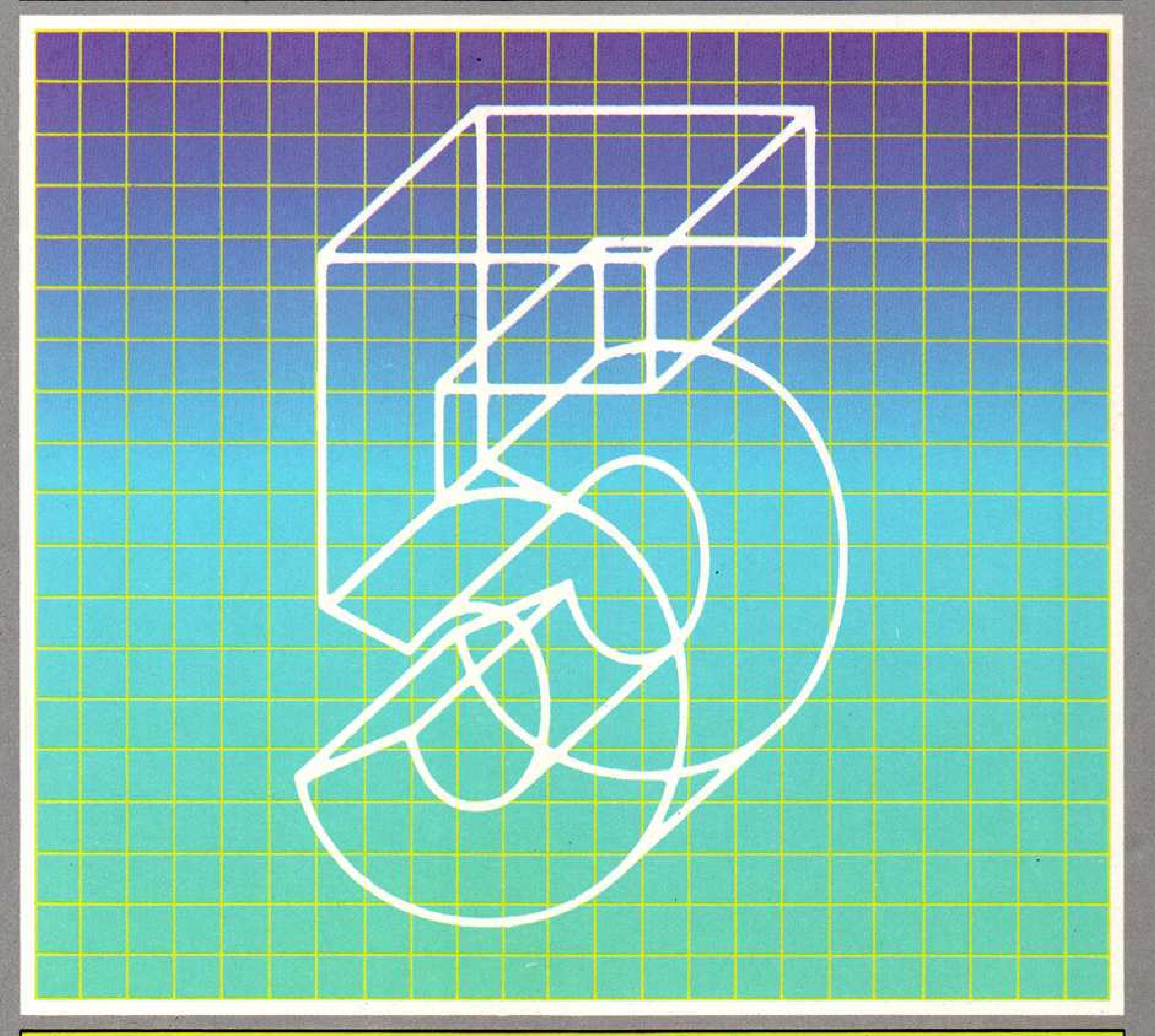
BIBLIOTECA PRACTICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

TALLER DE INFORMATICA

PROGRAMAS VALIDOS PARA AMSTRAD, IBM, SPECTRUM, COMMODORE Y MSX



PROGRAMAS PRACTICOS

TRUCOS, BRICOLAGE Y GRAFICOS

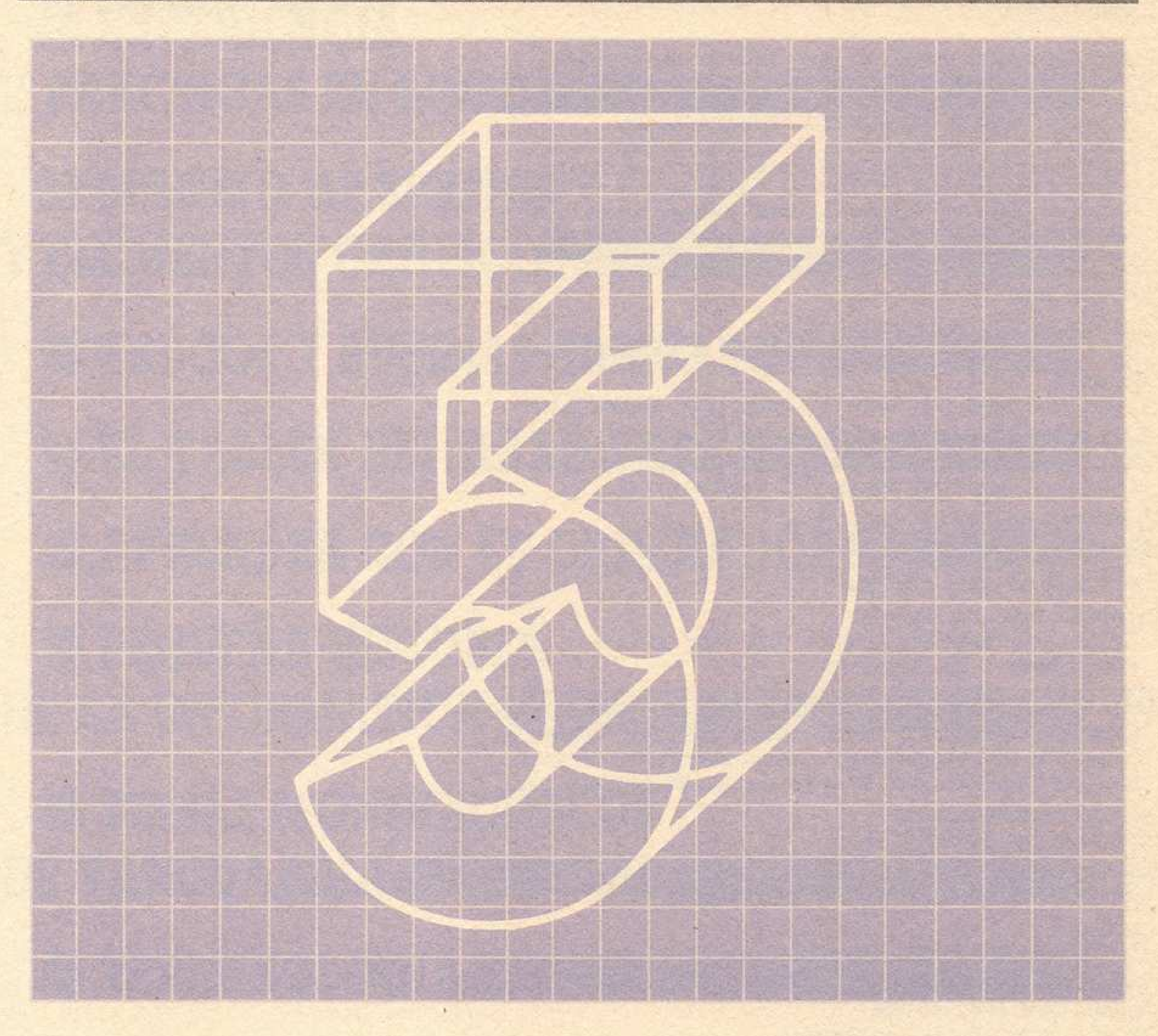
EDICIONES SIGLO CULTURAL

BIBLIOTECA PRACTICA

TALLER DE INFORMATICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD, IBM, SPECTRUM, COMMODORE Y MSX



EDICIONES SIGLO CULTURAL

Una publicación de

EDICIONES SIGLO CULTURAL, S. A.

Director-editor:
RICARDO ESPAÑOL CRESPO.

Gerente:
ANTONIO G. CUERPO.

Directora de producción:
MARIA LUISA SUAREZ PEREZ.

Director de la colección:
JOSE ARTECHE, Ingeniero de Telecomunicación

Diseño:
BRAVO-LOFISH.

Maquetación:
D. S.

Dibujos:
JOSE OCHOA

Tomo 5. «Experiencias prácticas en logo», Carlos Albert, Técnico de Informática. «Manejo de sprites y elementos gráficos», «Trucos y rutinas básicas», Francisco Morales, Técnico de Informática. «Aprende con el ordenador», AULA DE INFORMATICA APLICADA: Soledad Tamariz-Martel, Diplomada en Telecomunicación; Francisco Blanca, Diplomado en Telecomunicación; Alejandro Marcos, Licenciado en Química; Fernando Suero, Diplomado en Telecomunicación. «El Taller del Hardware», Carlos Rey, Ingeniero Industrial. «Pequeña historia de la Informática», «Temas monográficos de vanguardia», «Terminología», «Vocabulario de Informática», Blanca Arbizu, Técnico de Informática.

Ediciones Siglo Cultural, S. A.

Dirección, redacción y administración:
Sor Angela de la Cruz, 24-7.° G. Teléf. 279 40 36. 28020 Madrid.

Publicidad:
Gofar Publicidad, S. A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:
COEDIS, S. A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-055-3
ISBN de la obra: 84-7688-047-2

Fotocomposición:
ARTECOMP, S. A. Albarracín, 50. 28037 Madrid.

Imprime:
MATEU CROMO. PINTO (Madrid).

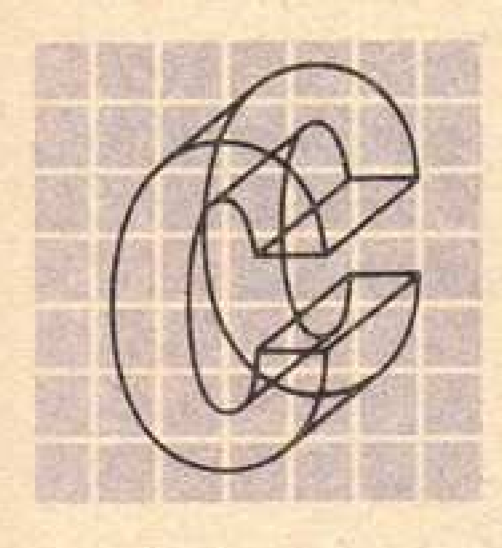
Depósito legal: M-43.185-1986
Printed in Spain - Impreso en España.

Suscripciones y números atrasados:
Ediciones Siglo Cultural, S. A.
Sor Angela de la Cruz, 24-7.° G. Teléf. 279 40 36. 28020 Madrid.

Enero, 1987.

INDICE

EXPERIENCIA Y PRACTICAS EN LOGO



CON unos ejemplos simples vamos a profundizar en algunas primitivas del LOGO.

Con las siguientes órdenes la Tortuga dibujará cuatro figuras geométricas. Las posiciones iniciales de cada figura las conseguimos con las órdenes:

PONX n
PONY n

Cambiamos la paleta de color con la primitiva:

PONPALETA n (n = 0 o n = 1)

FIGURAS GEOMETRICAS

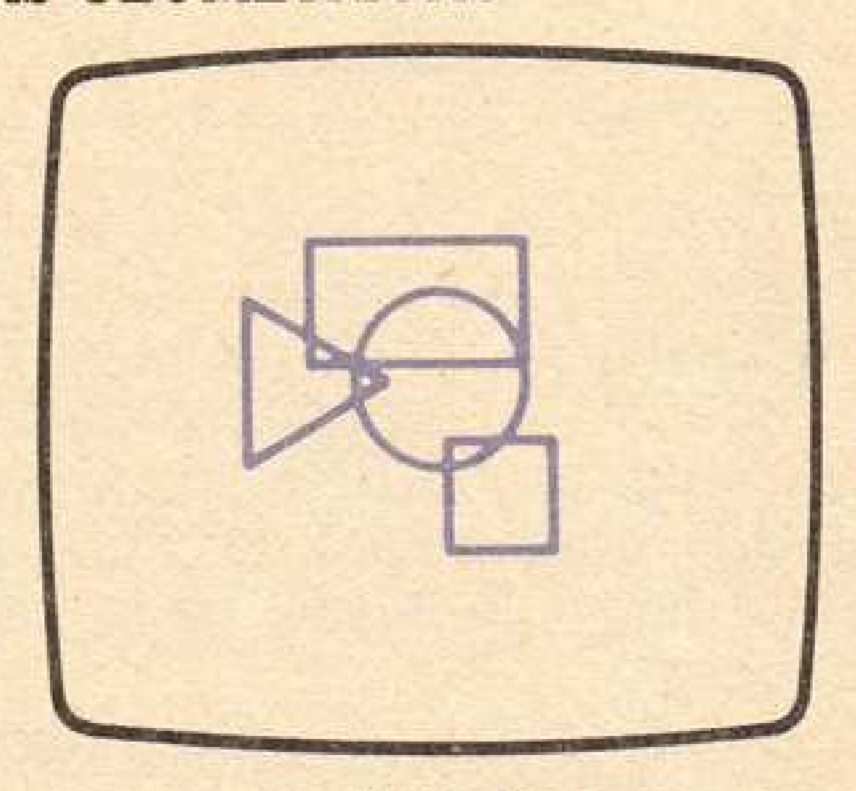


Figura 1.

INICIALIZACION

```
? PM
? BP
? SL
? PONCL 2
? OT
```

DIBUJOS FIGURAS

```
? PONX -50 PONY 50 BL
? REPITE 2 [GD 90 AV 100 GD 90 AV 60]
? SL PONX -60 PONY 15 BL
? REPITE 3 [GD 120 AV 80]
? SL PONX -20 PONY -5 BL
? REPITE 36 [AV 6 GD 10]
? SL PONX 10 PONY -20 BL
? REPITE 4 [GD 90 AV 70]
```

RELLENADO DE LAS FIGURAS

```
? SL PONX -40 PONY 40 BL
? PONCL 3
? RELLENA
? PONPALETA 0
? PONCL 2
? SL PONX -50 PONY -20 BL
? RELLENA
? PONCL 1
? SL PONX 18 PONY -1 BL
? RELLENA
```

Puedo variar el sentido de orientación de la Tortuga con otra nueva orden: PONRUMBO.

? PONPALETA 1
? PONCL 2
? SL PONX 30 PONY -50 BL
? RELLENA

NOTA: Si tu versión LOGO no dispone de la primitiva PONPALETA, no teclees la orden.

El siguiente ejemplo dibuja en pantalla un cuadrado, que vamos rellenándolo de diferentes colores.

Utilizamos aquí la orden ESPERA n para que observes que, tras rellenar la figura de un color, la Tortuga está unos segundos sin actuar; fíjate que la orden en cuestión está después de RELLENA.

INICIALIZACION

? PM
? BP
? SL
? PONCL 2
? OT

DIBUJO

? PONX -100 PONY 40 BL
? REPITE 4 [GD 90 AV 100]

RELLENADO

? SL PONX -80 PONY 20 BL
? REPITE 10 [PONCL AZAR 4 RELLENA ESPERA 15]

Cuando se ejecuten estas órdenes nosotros no sabremos qué sucesión de colores aparecerán, ya que no hemos fijado los colores, sino que los seleccionamos al azar.

Cuando la velocidad de algún proceso que realizamos resulta demasiado rápida, no tenemos tiempo de observar algo con detenimiento, podemos intercalar en el programa la orden ESPERA n, donde n serán pulsaciones de reloj, y deberá ser un número entero o bien será redondeado.

? ESPERA 1092

Si compruebas el tiempo transcurrido en tu reloj verás que 1092 pulsaciones de reloj corresponden a un minuto.

? BP BL PONCL 1
? REPITE 4 [AV 90 GD 90]
? ESPERA 100 BP
? REPITE 36 [AV 7 GD 10]
? ESPERA 200 BP

Ya hemos utilizado en varias ocasiones dos órdenes con las que podíamos posicionar a la Tortuga en cualquier zona de la pantalla sin tener que variar su orientación y sin tener que hacerla avanzar o retroceder.

Se trata de las órdenes:

PONX y PONY

Con estas dos órdenes podíamos variar la posición de la Tortuga, pero bien en una dirección o en otra alternativamente. Es decir, podíamos primero desplazarla en el sentido de las abscisas y luego en el sentido de las ordenadas, por ejemplo.

Existe otra orden en la cual se dan las coordenadas *x* e *y*, a la vez. Esta orden es:

PONPOS [X Y]

PONPOS es la abreviatura de PON POSICION.

Si, por ejemplo, damos la orden:

PONPOS [90 40]

Nos aparecerá lo siguiente:

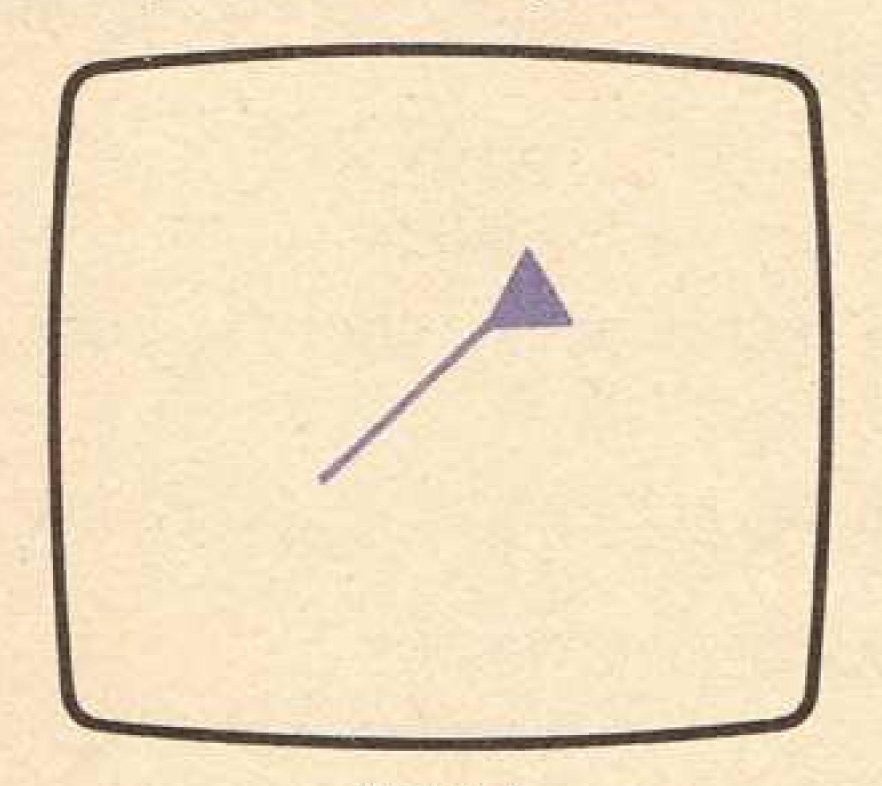


Figura 2.

La Tortuga se posiciona en las coordenadas indicadas, y a pesar de que se ha movido en diagonal, su rumbo no ha variado.

Si hasta ahora, cuando queríamos posicionar la Tortuga en una determinada posición de la pantalla, utilizábamos dos órdenes, ahora podemos hacer lo mismo, pero utilizando sólo una orden.

Si en esta orden se utilizan decimales, los dos primeros se toman en cuenta y el tercero será redondeado.

En una sola orden puedo englobar más de una. Este es el caso de la orden PONLAPIZ.

Recuerda que la Tortuga, si está en la posición central, tiene de coordenadas (0,0) y que los desplazamientos hacia la derecha, en el sentido de las x, o hacia arriba en el sentido de la y, serán positivos. Hacia la izquierda en x, o hacia abajo en y, serán negativos.

Teclea e introduce las siguientes órdenes y lo verás:

INICIALIZACION

? PM
? BP
? OT
? PONFONDO 1
? PONCL 2

DIBUJO

? SL PONPOS [-159 124]
? BL PONPOS [160 125]
? SL PONPOS [-159 125]
? BL PONPOS [160 -124]

Obtendrás una pantalla como ésta:

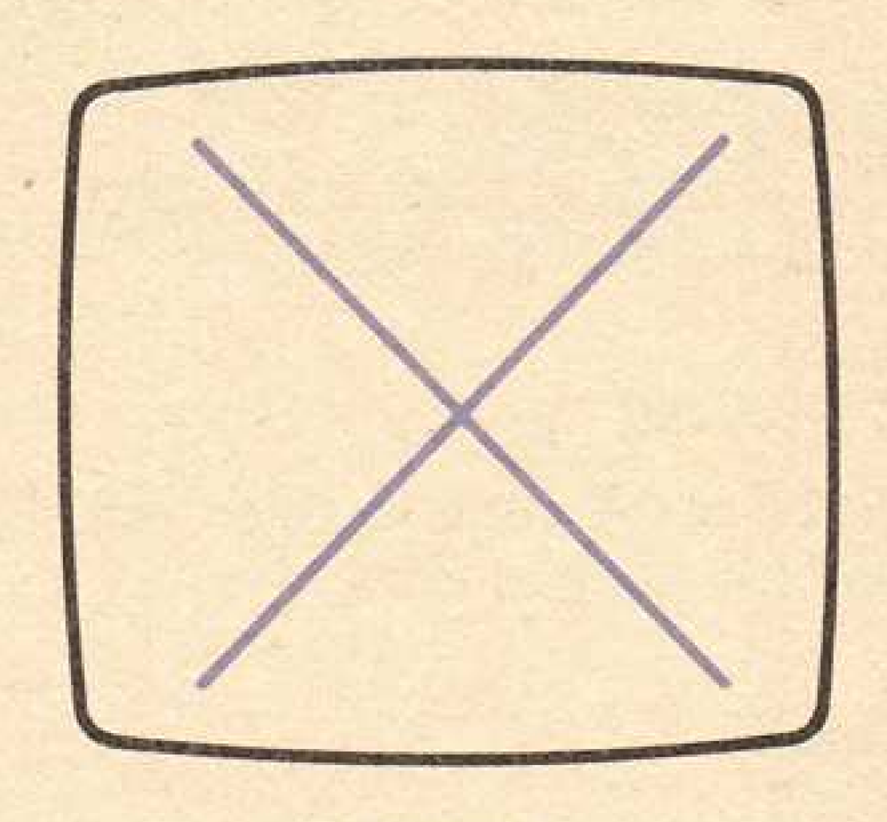


Figura 3.

En este caso, los valores que hemos dado a la orden PONPOS sólo son válidos si posees un ordenador que tenga una resolución gráfica de 320 × 250 puntos o pixels. Si no es así, cambia estos valores limitándote a la resolución de tu ordenador.

Si cuando introducimos la orden PONPOS [X Y] tenemos el lápiz alzado, nos situamos en la posición elegida sin dejar rastro; si lo bajamos, dejaremos una línea pintada desde donde nos encontrábamos hasta el punto elegido [X Y].

Ya sabemos acceder a cualquier punto de la pantalla dando las coordenadas correspondientes. Cambiemos de rumbo.

Introduce las siguientes órdenes:

INICIALIZACION

? BP BL
? PM MT
? PONFONDO 1
? PONCL 2

PUNTOS CARDINALES

? PONRUMBO 90 AV 100
? CENTRO
? PONRUMBO 180 AV 100
? CENTRO
? PONRUMBO 270 AV 100
? CENTRO
? PONRUMBO 360 AV 100
? CENTRO
? PONCL 3
? PONRUMBO 0 AV 100

La Tortuga ha ido dibujando hacia los cuatro puntos cardinales; con rumbo 90 hacia el este, rumbo 180 hacia el sur, rumbo 270 hacia el oeste y rumbo 360 hacia el norte.

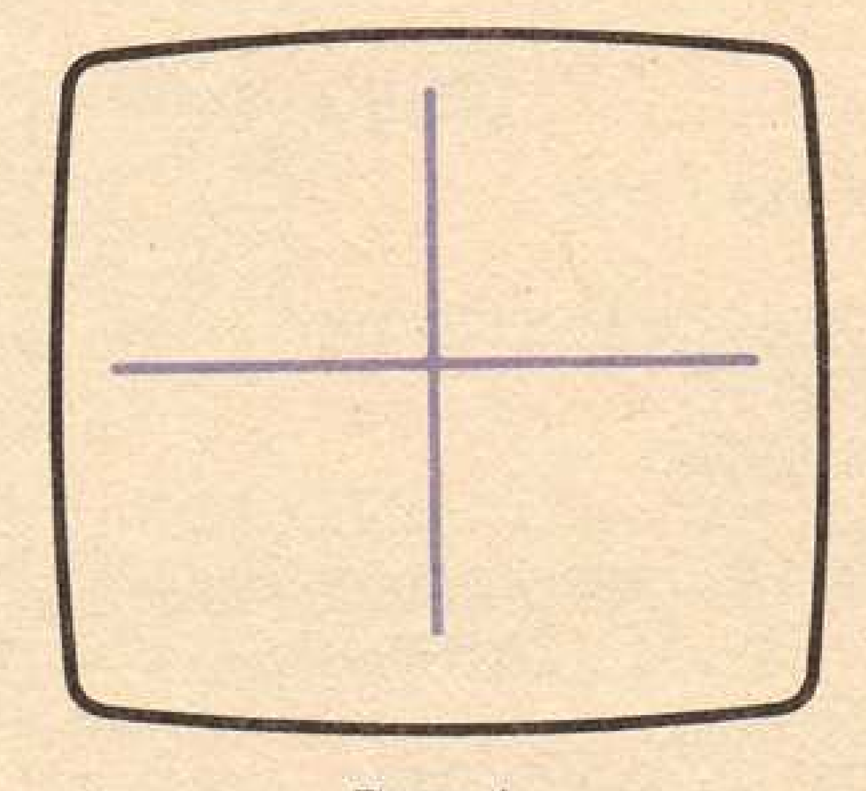


Figura 4.

PONRUMBO n

Practicando con esta orden apreciarás la diferencia existente con las órdenes GD n y GI n, ya que el movimiento de la Tortuga con estas dos órdenes es relativo respecto a su orientación; en cambio, con PONRUMBO el movimiento de la Tortuga es absoluto.

Aquí tienes dos ejemplos que te ayudarán a ver mejor cómo funciona esta nueva orden:

Puedo dar diferentes colores al lápiz, al fondo, al texto y al fondo del texto.

LINEAS RADIALES

INICIALIZACION

? PM
? BP
? OT
? PONFONDO 1

DIBUJO

? REPITE 100 [PONRUMBO AZAR 360 PONCL AZAR 4 AV AZAR 100 CENTRO]

Con estas órdenes obtenemos un conjunto de líneas que parten todas del centro. Cada una de ellas tiene tanto una longitud como un color elegido al azar. De igual manera, cada una parte del centro hacia una dirección que desconocemos, ya que se determina el rumbo de la Tortuga de una forma aleatoria, pudiendo oscilar entre 0 y 359 grados.

INICIALIZACION

? PM
? BP
? OT
? PONFONDO 0
? SL

DIBUJO

? PONX -58
? BL
? REPITE 36 [AV 10 GD 10]
? SL CENTRO
? BL RELLENA
? PONCL 3
? REPITE 50 [PONRUMBO AZAR 360 AV 50 CENTRO]
? PONCL 2
? REPITE 50 [PONRUMBO AZAR 360 AV 40 CENTRO]

PINTAR PUNTOS

PUNTO [X Y]

Ejemplo:

PUNTO [80 62]
PUNTO [80 -62]
PUNTO [-80 62]
PUNTO [-80 -62]

Los puntos se pintan en la pantalla sin que la Tortuga se mueva.

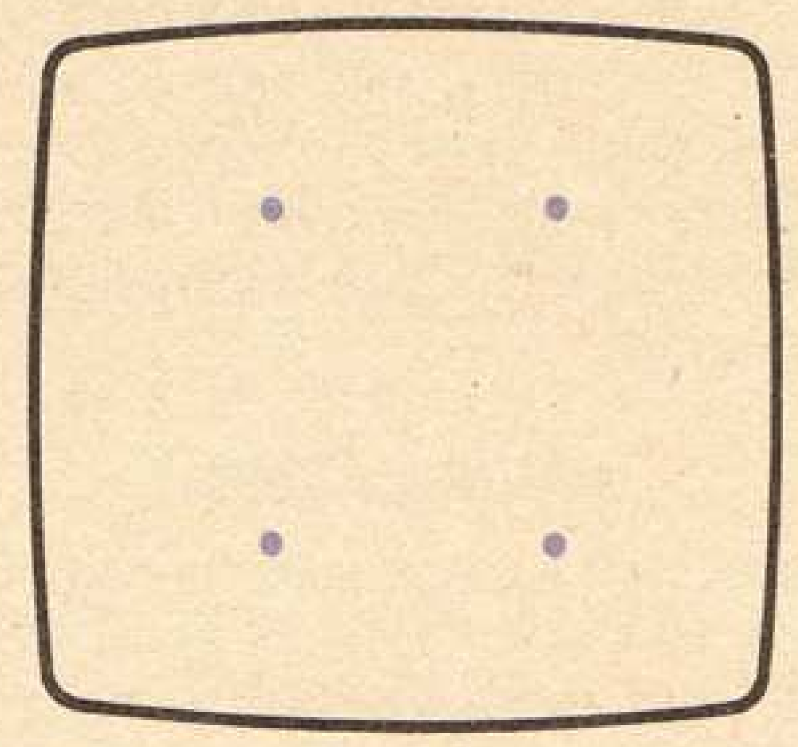


Figura 5.

Recordemos que el punto [0 0] es el centro de la pantalla.

Si damos a X un valor superior a -159 ó 160 el punto aparecerá o no marcado, dependiendo del estado en que se encuentra la pantalla. Recuerda las primitivas: ENLAZA, LIMITA y VENTANA.

Lo mismo ocurrirá con los valores -124 y 125 de la coordenada Y.

Estos valores de X e Y que hemos especificado son válidos en pantallas con resolución 320 x 250 puntos.

Tanto para el valor de X como Y se tienen en cuenta dos decimales.

? PONCL 2
? PUNTO [30 50]

El color del punto dibujado es el que en ese momento tiene el lápiz.

? BP
? SUBELAPIZ
? PONCL 3
? PUNTO [50 40]

Observa que, aunque el lápiz está alzado, el punto se ha marcado.

? BP
? PONLAPIZ [SL 3 1]
? PUNTO [50 40]

En este caso hemos introducido una nueva orden:

PONLAPIZ

Esta orden admite tres parámetros que deben ir encerrados entre corchetes. Estos son:

Algunas versiones me permiten cambiar el color de la Tortuga.

— Estado del lápiz (SL, BL, IL, GOMA).
— Color del lápiz.
— Paleta de color.

Esta orden nos permite englobar a tres en una. Así nos evitamos el tener que dar más órdenes de lo necesario.

La siguiente orden:

PONLAPIZ [SL 2 0]

Equivale a las órdenes:

```
SL
PONCL 2
PALETA 0
```

Por ejemplo, con la orden:

PONLAPIZ [SL 1 1] AV 100

Avanzamos sin pintar.

Y con la orden:

PONLAPIZ [BL 1 0] AV 100

Pintamos una línea.

Esta orden no la poseen todas las versiones del LOGO. Sí existe en el LOGO de los PC-compatibles.

Aquí tienes las órdenes que dibujan una habitación en tres dimensiones gracias a un cierto sentido de profundidad que le hemos dado.

INICIALIZACION

```
? PM
? BP
? SL
? OT
? PONCL 1
```

PRIMER PLANO

```
? PONPOS [100 -30]
? BL
? REPITE 2 [AV 120 GI 90 AV 200 GI 90]
```

SEGUNDO PLANO

```
? SL
? PONPOS [32 15]
? BL
? REPITE 2 [AV 30 GI 90 AV 65 GI 90]
? PONPOS [100 -30]
? AV 120 PONPOS [32 45]
? GI 90 AV 65
? PONPOS [-100 90]
? GI 90 AV 120
? PONPOS [-33 15]
```

ALFOMBRA

```
? SL PONPOS [-20 15]
? BL PONPOS [20 5] PONPOS [55 -20]
? PONPOS [-55 -20] PONPOS [-20 5]
? SL PONPOS [-17 2]
? BL PONPOS [17 2] PONPOS [44 -17]
? PONPOS [-44 -17] PONPOS [-17 2]
? SL AV 4 BL
? PONCL 2 RELLENA
```

PARED TRASERA

```
? SL RE 40 BL
? PONCL 15
? RELLENA
? PONCL 13 AV 10
? SL GI 90 AV 35
? BL GI 90 AV 9
```

CUADRO

```
? PONCL 6 SL
? PONPOS [-80 10]
? BL
? PONPOS [-80 50]
? PONPOS [-50 35]
? PONPOS [-50 20]
? PONPOS [-80 10]
? SL GD 45 AV 4
? RELLENA
```

Si quieres, puedes ir completando la habitación dibujando alguna cosa más en su interior. Puedes también cambiar los colores y las dimensiones.

Colores II

Con respecto al color, ya vimos dos de las órdenes más importantes que el Logo posee. Eran PONCL y PONFONDO. Con ellas cambiamos el color tanto del lápiz de la Tortuga como el del fondo de la pantalla. Vimos también los cuadros con los códigos de colores que poseen diferentes ordenadores.

Hay órdenes como PUNTO, que no las realiza la Tortuga.

En esta ocasión vamos a completar el tema del color, explicando otra serie de órdenes que el Logo posee.

Existe otra orden que nos permite cambiar el color del texto; se trata de la orden PONCOLORTEXTO, y su forma abreviada es PONCT.

Dependiendo del ordenador, esta orden presenta algunas diferencias. Vamos a verla en particular para cada uno de ellos.

PC-COMPATIBLES

PONCT n

Pone el color al texto que determina el valor de n.

Esta orden puede escribirse tanto de la forma completa como de la forma abreviada:

MSX

PONCT n
PONCT lista

Si la orden se da de la forma PONCT n, el color del texto aparecerá, desde el momento que se da la orden, del color que determina n.

Si en lugar de dar sólo un valor damos dos, es decir, utilizamos la segunda forma, el primer valor nos determinará el color del texto y el segundo el color del fondo del texto.

Esta orden sólo puede darse en este caso, de forma abreviada.

SPECTRUM

PONCT n m

El color del fondo del texto será el que determina n, y el color del texto será el que determina m.

En este ordenador, la orden se tiene que dar también de la forma abreviada.

Otras órdenes de color

Existe en los PC-compatibles una forma para dar color al fondo del texto, al igual que lo hacíamos en los otros dos casos. Esto lo conseguimos con la orden:

PONFONDOTEXTO n o bien
PONFT n

Esta orden pone el color del fondo de los caracteres que aparecen en la pantalla, según el valor de n.

En los MSX podemos variar el color de la forma de la Tortuga si damos la orden:

PONCF n

Dependiendo del valor de n, la Tortuga nos aparecerá de un color o de otro.

En los SPECTRUM podemos variar el color del borde de la pantalla. Esto lo conseguimos con la orden:

PONCOLORBORDE n

o bien

PONCB n

El borde se pondrá del color que viene dado por n.

En todas estas órdenes que hemos tratado, el valor de n ha de ser uno que corresponda al cuadro de código de colores que vimos anteriormente.

Mensajes Logo

— NO SE QUE DEBO HACER CON o DIME QUE HACER CON.

No se ha precisado el dato que debe recibir la orden o sentencia especificada.

— FALTAN DATOS PARA o FALTAN DATOS EN.

Indica que la orden escrita precisa dar más datos para ejecutarse.

— ERROR EN EL SISTEMA LOGO.

La carga del Logo no ha sido correcta. Hay que repetir la carga.

— ALTO o INTERRUMPIDO.

Los mensajes que el LOGO me da varían según el ordenador y la versión con la que estemos trabajando.

Se ha detenido la ejecución de un proceso.

— ... ES UNA PRIMITIVA.

Se ha utilizado como dato el nombre de alguna primitiva.

— PAUSA.

Por pulsar F5 la ejecución se detiene. Se puede continuar tecleando CONTINUA. (Este mensaje sólo aparece en las versiones de algunos ordenadores en concreto, como por ejemplo, en los PC-compatibles.)

— PONCOLORBORDE n.

PONCB n.

Pone el borde de la pantalla del color que determina n. (Válido para ordenadores Spectrum.)

— PONCF n.

Pone del color que determina n, el color de la forma de la Tortuga que tenemos definida. (Válido para ordenadores MSX.)

Cuadro resumen

— PONFOS [X Y].

La Tortuga se sitúa en el punto de coordenadas que determinan los valores de X e Y.

— PONRUMBO n.

La orientación de la Tortuga variará hacia la dirección que determina n, que vendrá expresada en grados.

— PUNTO [X Y].

Se dibuja un punto en la posición que determinan las coordenadas X e Y permaneciendo la Tortuga en el lugar en que se encontraba.

— PONLAPIZ [estado lápiz color paleta].

Define el estado del lápiz (SL, BL, IL o GOMA), el color y la paleta, respectivamente.

— PONCOLORTEXTO n.

PONCT n.
PONCT lista.
PONCT n m.

Pone del color que determina n el texto, si sólo se especifica un valor. Si se especifica un segundo valor, corresponderá al color del fondo del texto.

— PONFONDOTEXTO n.

PONFT n.

Pone del color que determina n, el fondo del texto. (Válido para ordenadores PC-compatibles.)

Ejercicios

1. *Utilizando la orden* PONFOS [X Y], *intenta hacer el siguiente dibujo:*

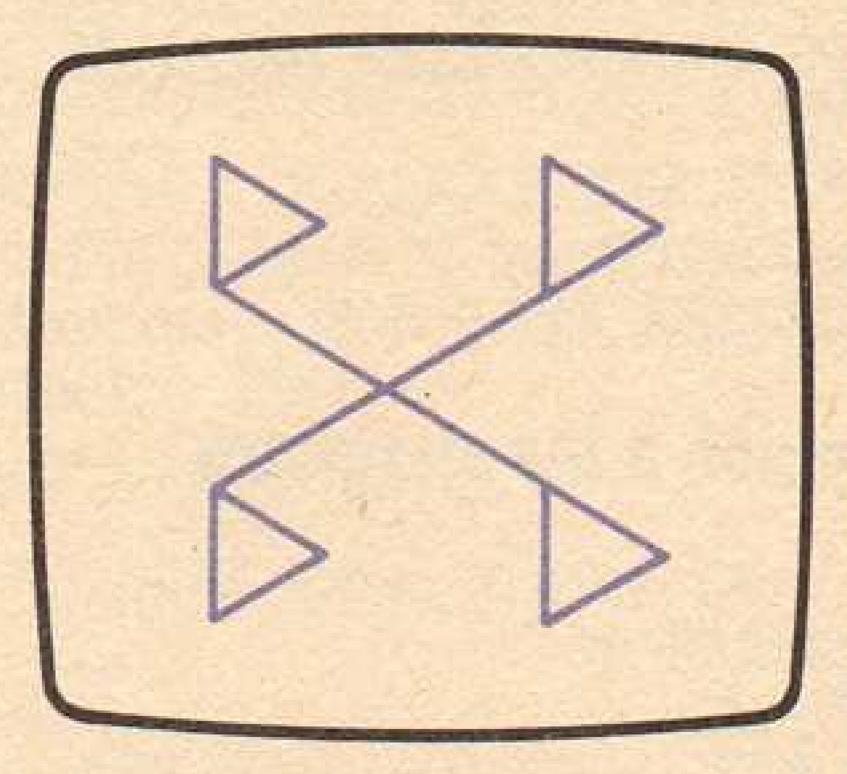


Figura 6.

2. *¿Hubiese aparecido el mismo dibujo, si en vez de dar la orden GD 120 hubiésemos dado la orden* PONRUMBO 120*?*
3. *Practica con la orden* PONRUMBO *dibujando la siguiente figura:*

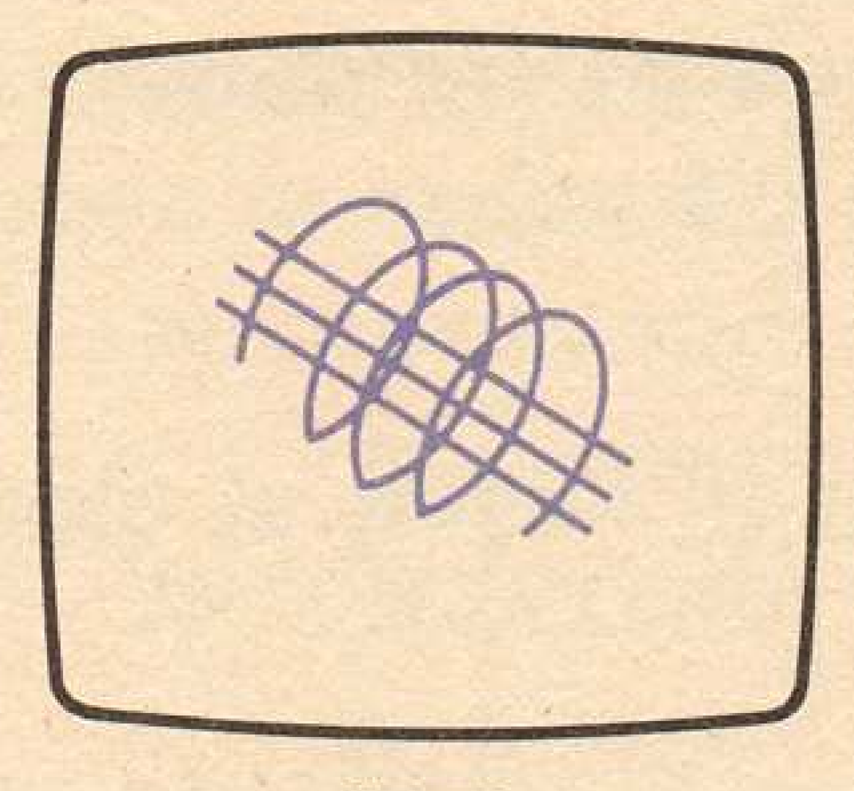


Figura 7.

Dependiendo del ordenador, la resolución gráfica es distinta.

4. *Haz el siguiente dibujo:*

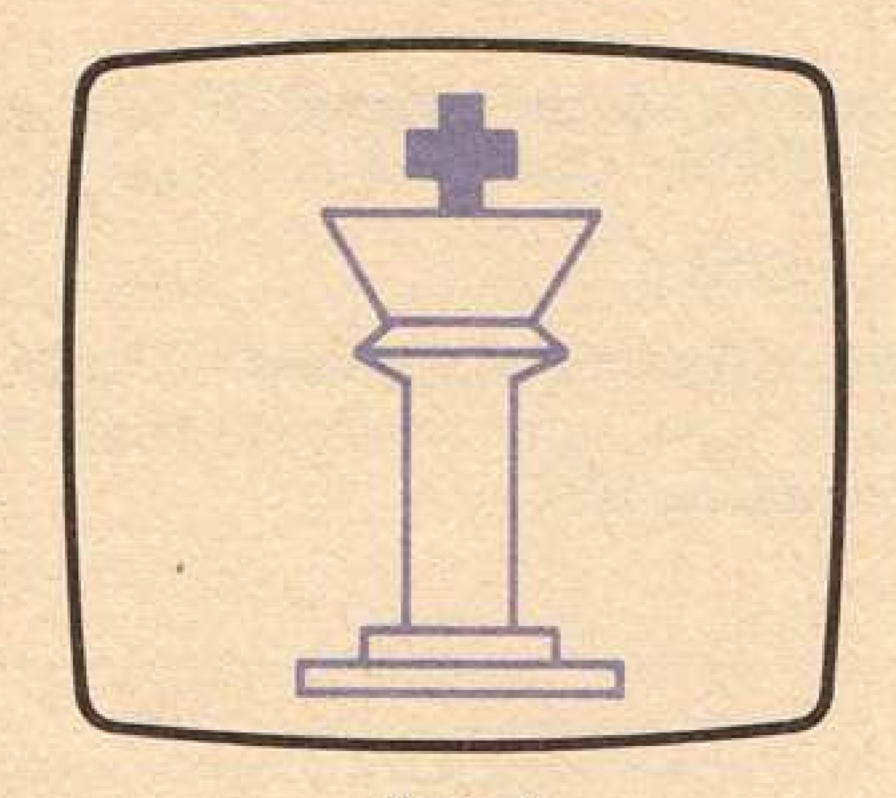


Figura 8.

5. *Son correctas las siguientes órdenes:*

— PONPOS [AZAR 100 30]
— PUNTO (90 20)
— PONRUMBO AZAR 200 AV 40 GD 60 RE 100
— PONLAPIZ [GOMA 2]
— PONCT 3
— ESPERA AZAR 3000

Solución a los ejercicios

1:

INICIALIZACION
? PM
? BP
? OT
? SL
? PONFONDO 9
? PONCL 2

DIBUJO TRIANGULOS
? PONPOS [70 52]
? BL
? REPITE 3 [AV 40 GD 120]
? SL
? PONPOS [70 -52]
? BL
? REPITE 3 [AV 40 GD 120]
? SL
? PONPOS [-70 52]
? BL
? REPITE 3 [AV 40 GD 120]
? SL
? PONPOS [-70 -52]
? BL
? REPITE 3 [AV 40 GD 120]

TRAZADO DE LAS LINEAS
? SL
? PONPOS [70 52]
? BL
? PONPOS [-70 -12]
? SL
? PONPOS [-70 52]
? BL
? PONPOS [70 -12]

2:

No, ya que pintaríamos siempre en la misma dirección. La Tortuga estaría siempre orientada hacia la misma dirección (120 grados) y lo que se tiene que conseguir es que, después de avanzar, gire 120 grados.

3:

INICIALIZACION
? PM
? BP
? SL
? OT
? PONCL 1

DIBUJO DE LA ESPIRAL
? PONPOS [-100 50]
? BL
? REPITE 4 [REPITE 27 [AV 4 GD 10] GD 90]

DIBUJO DE LAS TRES LINEAS
? SL
? PONPOS [12 -35]
? BL
? PONRUMBO 310
? AV 160 SL
? PONPOS [24 -35]
? BL
? PONRUMBO 310
? AV 160 SL
? BL
? PONRUMBO 310
? AV 160

4:

INICIALIZACION
? PM

Hay ordenadores en los que sólo se pueden dar dos colores por cada matriz de 8 × 8 puntos.

```
? SL
? BP
? OT
```

CENTRANDO DIBUJO

```
? GI 90 AV 35
? GD 90 RE 60
? BL
```

DIBUJO

```
? REPITE 2 [AV 10 GD 90 AV 70 GD 90]
? GI 90 RE 5 GD 90
? SL AV 10 BL
? AV 5 GD 90 AV 60
? GD 90 AV 5 RE 5
? GD 90 AV 15 GD 90
? AV 65 GD 45 AV 5
? GI 135 AV 37 RE 37
? GD 70 AV 10 GD 20
? AV 2 GI 90 AV 28
? RE 28 GD 115 AV 30
? SL CENTRO
? RE 45 GI 90 AV 15
? GD 90 BL AV 65
? GI 45 AV 5 GD 70
? AV 10 GI 20 AV 2
? GI 30 AV 30 GD 115
? AV 55
```

CRUZ

```
? RE 30 GI 90 AV 10
? GI 90 AV 5 GD 90
? AV 5 GD 90 AV 5
? GI 90 AV 7 GD 90
? AV 5 GD 90 AV 7
? GI 90 AV 5 GD 90
? AV 5 GD 90 AV 5
? GI 90 AV 10
? RE 4 GD 90 SL
? AV 2 BL RELLENA
```

5:

- PONPOS [AZAR 100 30]: INCORRECTA. La orden PONPOS no admite como parámetro la orden AZAR.
- PUNTO (90 20): INCORRECTA. En lugar de paréntesis deben ir corchetes.
- PONRUMBO AZAR 200 AV 40 GD 60 RE 100: CORRECTA.
- PONLAPIZ [GOMA 2]: INCORRECTA. Falta un parámetro; esta orden debe contener siempre tres datos.
- PONCT 3: CORRECTA.
- ESPERA AZAR 3000: CORRECTA.

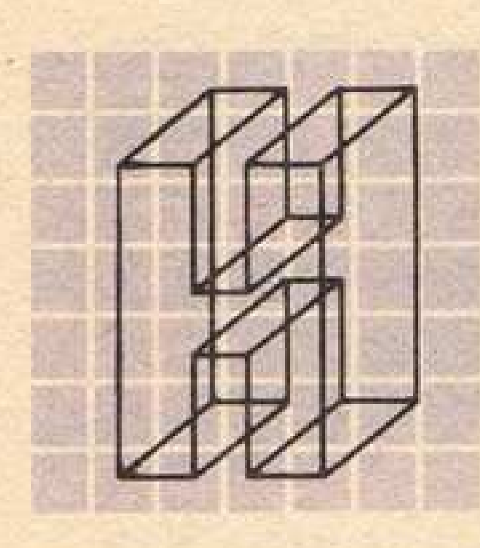
Si no posees un monitor o televisor en color, los colores aparecen como diferentes intensidades de los que tenga tu monitor.

En el Spectrum, la forma de ocultar la Tortuga no es con la orden OT, sino con la orden ET (esconde tortuga).

La orden PONPALETA sólo la posee el logo de los PC-compatibles.

En los ordenadores PC compatibles cuando doy la orden PM, a la vez que la pantalla se sitúa en el modo mixto, me aparece la Tortuga.

MOVIMIENTO DE FIGURAS COMPLEJAS



HASTA ahora sólo hemos hecho programas de movimiento con figuras compuestas de un solo carácter. Tal es el caso de los programas que vimos de la estrella fugaz o de la pelota que va rebotando contra las paredes.

En este tomo veremos cómo realizar movimientos con figuras compuestas por más de un carácter. Dichos caracteres pueden ser definidos por el usuario o bien caracteres ya predefinidos dentro del propio ordenador.

En tomos anteriores vimos que había diversos tipos de movimiento según la dirección y el sentido que llevase el carácter al moverse por la pantalla. Dichos tipos de movimiento son aplicables también a las figuras formadas por más de un carácter, pero al ser estas figuras más complicadas, el movimiento se vuelve más complejo.

Los tipos de movimientos posibles que vimos eran:

— Movimiento horizontal. De izquierda a derecha o de derecha a izquierda.
— Movimiento vertical. De arriba a abajo o de abajo a arriba.
— Movimiento diagonal.
— Movimiento circular.

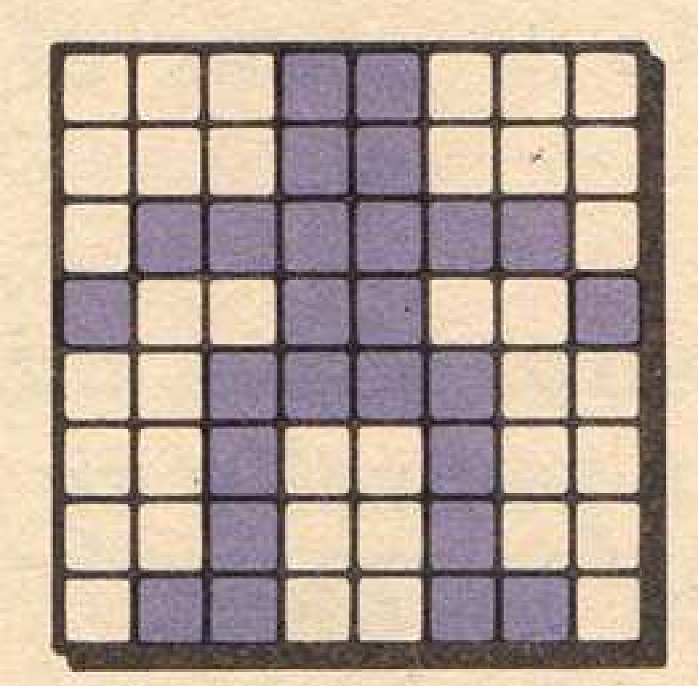


Fig. 1. Definición de un hombrecillo de un solo carácter.

— Movimiento uniformemente acelerado o decelerado.

Estudiaremos todos estos tipos de movimiento para figuras complejas, pero antes vamos a ver algunas figuras complejas que podemos dibujar en nuestro ordenador.

Cuando queremos hacer que un hombre se mueva por la pantalla tenemos dos posibles opciones. Una es definir dicho hombre en un solo carácter (ver fig. 1). La otra es definirlo mediante varios caracteres. De esta última manera el hombrecillo estará mejor definido y se parecerá más a un hombre de verdad (ver fig. 2).

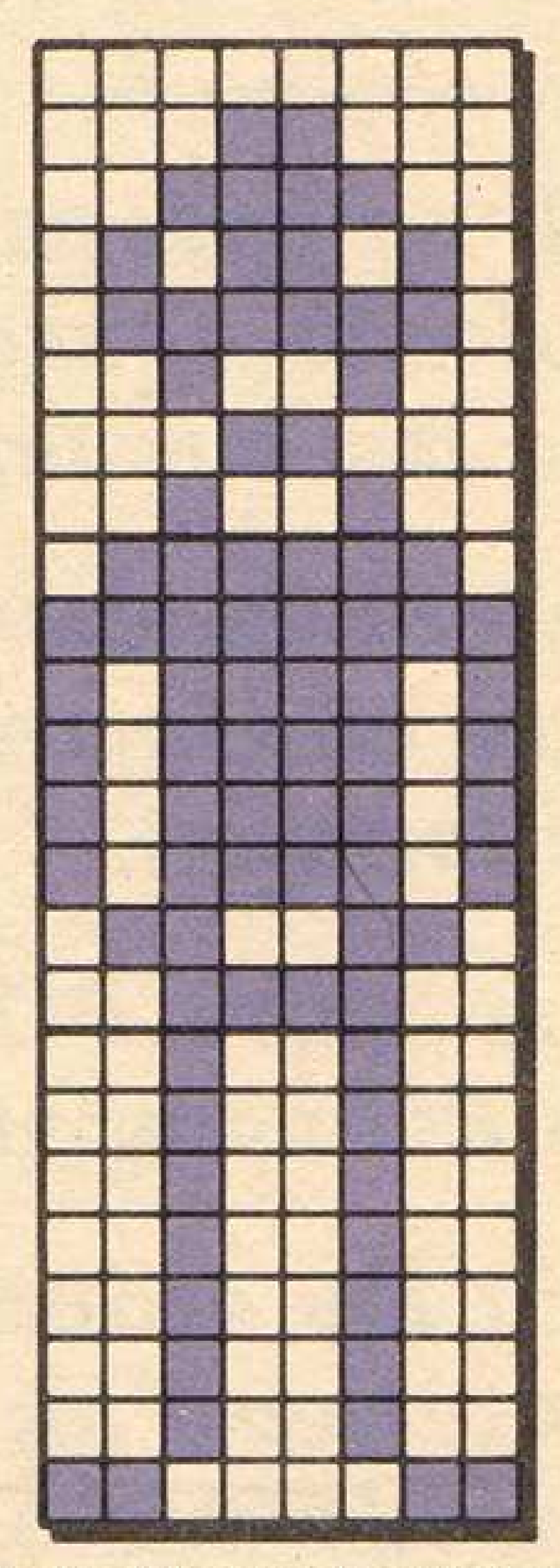


Fig. 2. Definición de un hombrecillo utilizando tres caracteres, uno debajo de otro.

Otra de las maneras de definir figuras más complejas es utilizando los caracteres que ya tiene definidos el ordenador. Conjugando varios de dichos caracteres, se pueden crear figuras bastante complejas y vistosas. Como ya vimos en tomos anteriores, todos los ordenadores tienen, aparte del grupo de caracteres normales, un grupo de caracteres llamados semigráficos. Con estos caracteres se pueden realizar figuras bastante buenas sin necesidad de tener que estar definiendo nuestros propios caracteres. En la figura 3 vemos algunas figuras que pueden realizarse con los caracteres normales del ordenador. En la número 4 vemos algunas figuras que pueden realizarse con los caracteres semigráficos.

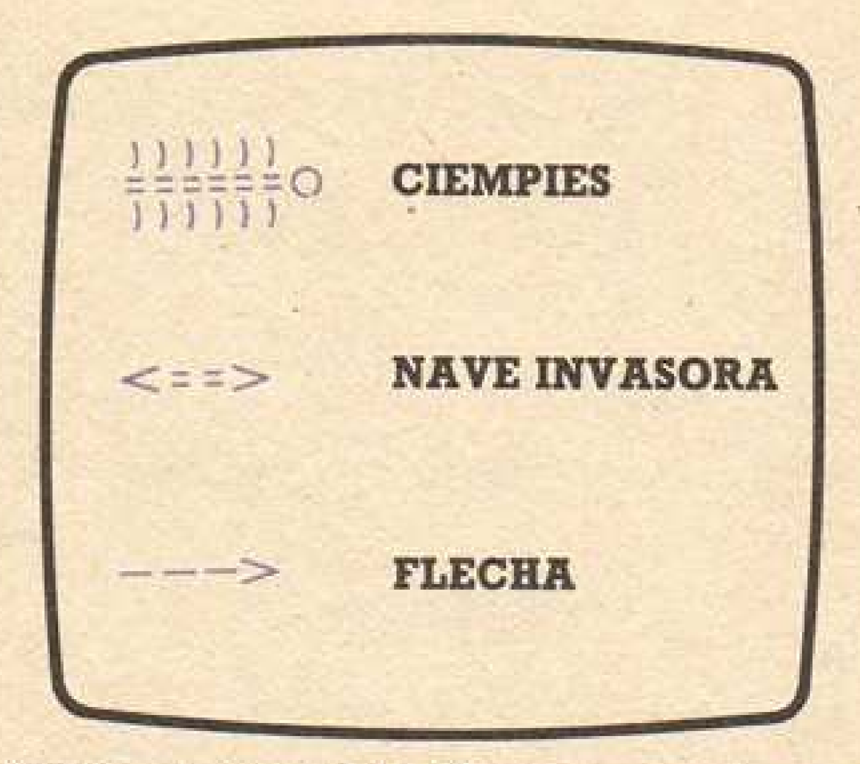


Fig. 3. Estas son algunas de las figuras que podemos definir con los caracteres de nuestro ordenador.

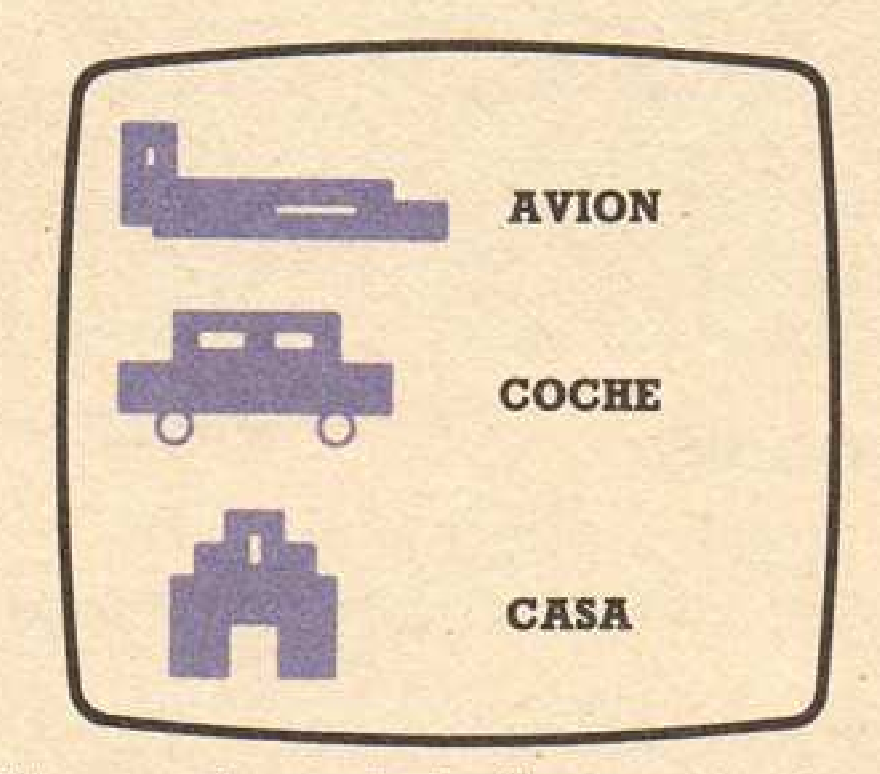


Fig. 4. Estas son algunas de las figuras que podemos dibujar con los caracteres semigráficos de nuestro ordenador.

Como los caracteres semigráficos varían de unos ordenadores a otros, en la figura 5 os mostramos las tablas de dichos caracteres en todos los ordenadores.

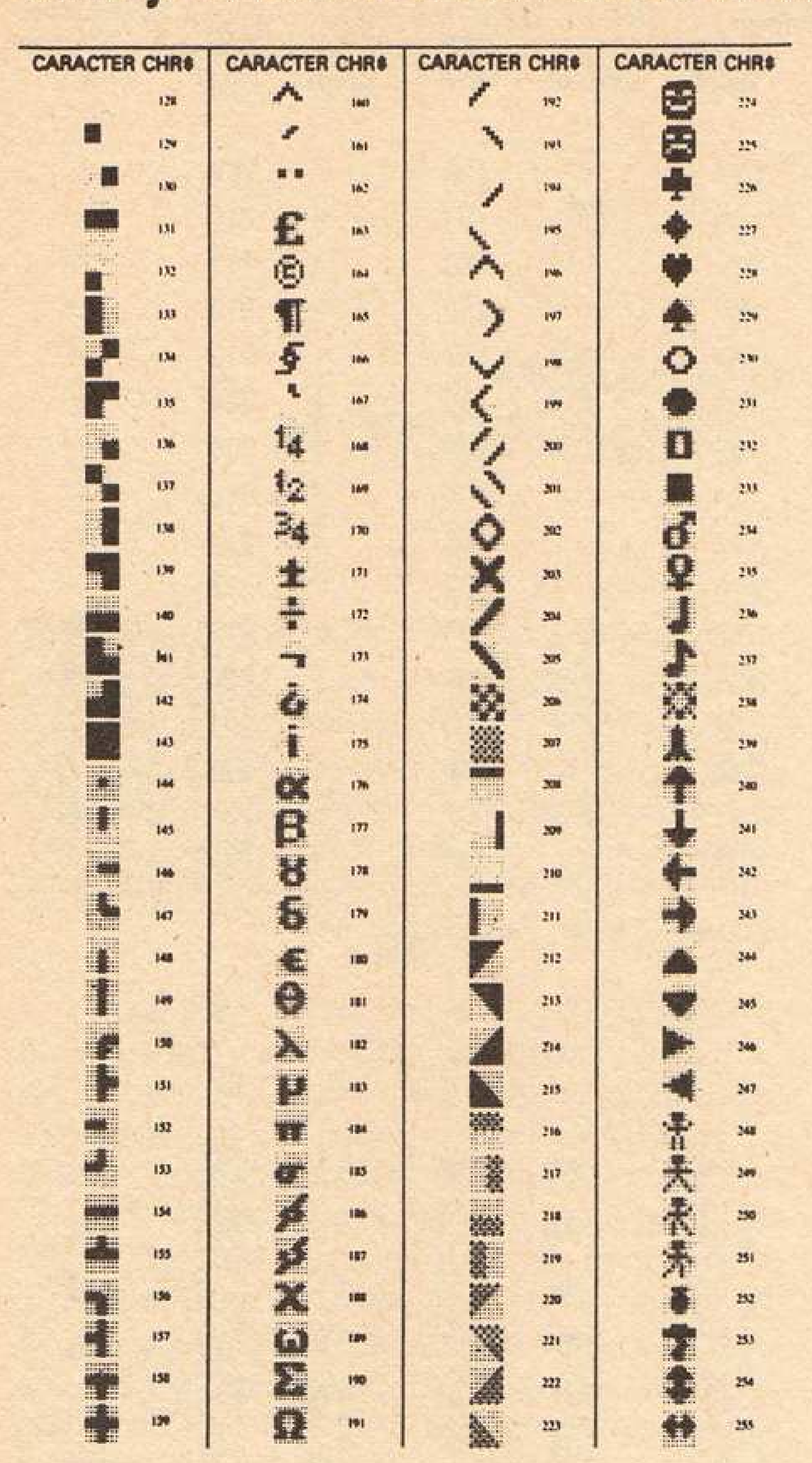
Una vez que hemos visto cómo son las figuras complejas y cómo construirlas, vamos a entrar de lleno en la materia que nos interesa.

Valor ASCII	Carácter	Valor ASCII	Carácter	Valor ASCII	Carácter
177	▒	193	┴	209	╤
178	▓	194	┬	210	╥
179	│	195	├	211	╙
180	┤	196	─	212	╘
181	╡	197	┼	213	╒
182	╢	198	╞	214	╓
183	╖	199	╟	215	╫
184	╕	200	╚	216	╪
185	╣	201	╔	217	┘
186	║	202	╩	218	┌
187	╗	203	╦	219	█
188	╝	204	╠	220	▄
189	╜	205	═	221	▌
190	╛	206	╬	222	▐
191	┐	207	╧	223	▀
192	└	208	╨		

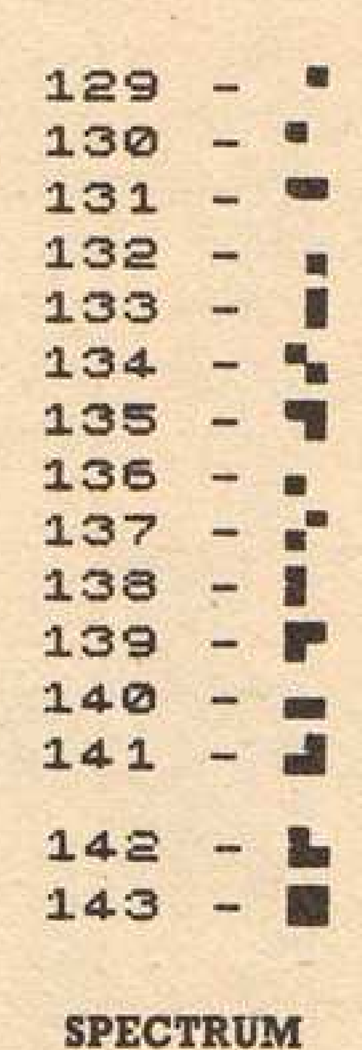
IBM

	97		115		173
	98		116		174
	99		117		175
	100		118		176
	101		119		177
	102		120		178
	103		121		179
	104		122		180
	105		123		181
	106		124		182
	107		125		183
	106		164		184
	107		165		185
	108		166		186
	109		167		187
	110		168		188
	111		169		189
	112		170		190
	113		171		191
	114		172		

COMMODORE



AMSTRAD



SPECTRUM

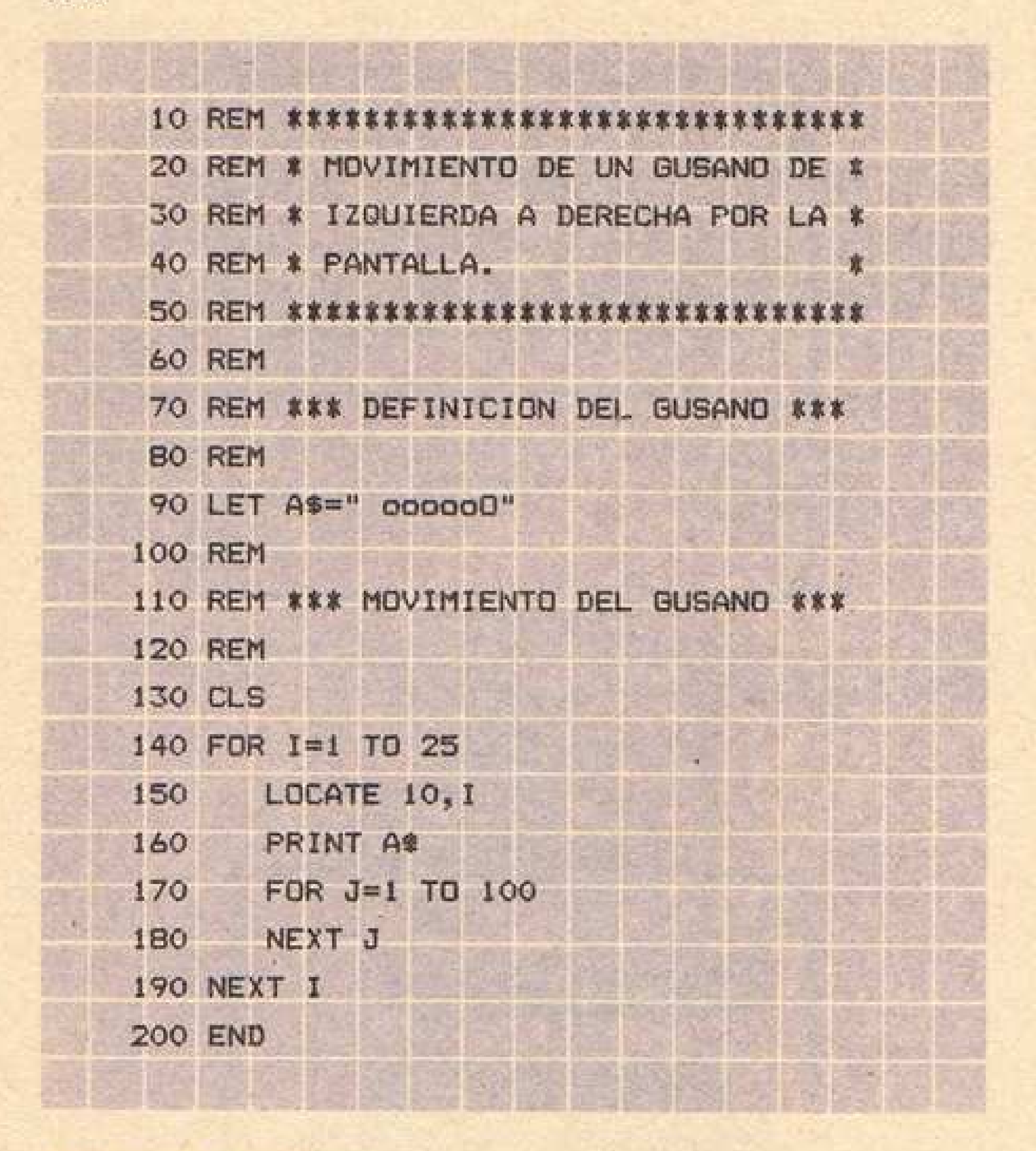
Fig. 5. Estos son los caracteres semigráficos de los distintos ordenadores.

Movimiento horizontal de figuras complejas

Este tipo de movimiento es casi igual que el que se vio para figuras compuestas de un solo carácter. La única diferencia estriba en que, como son más de un carácter los que configuran el objeto a mover, habrá que tener en cuenta todos y cada uno de los caracteres por separado.

El programa 1 nos muestra un movimiento horizontal de izquierda a derecha muy sencillo.

Las modificaciones necesarias para que este programa funcione en ordenadores distintos del IBM y del AMSTRAD son las siguientes:



```
 10 REM ********************************
 20 REM * MOVIMIENTO DE UN GUSANO DE *
 30 REM * IZQUIERDA A DERECHA POR LA *
 40 REM * PANTALLA.                  *
 50 REM ********************************
 60 REM
 70 REM *** DEFINICION DEL GUSANO ***
 80 REM
 90 LET A$=" oooooO"
100 REM
110 REM *** MOVIMIENTO DEL GUSANO ***
120 REM
130 CLS
140 FOR I=1 TO 25
150    LOCATE 10,I
160    PRINT A$
170    FOR J=1 TO 100
180    NEXT J
190 NEXT I
200 END
```

COMMODORE

130 PRINT "<SHIFT-HOME>"
150 LET Y=10:LET X=I:GOSUB 9500

Para que este programa funcione es necesario unirlo a la rutina LOCATE PARA COMMODORE que se dio en el tomo 1 y que volvemos a repetir en éste.

MSX	**SPECTRUM**
150 LOCATE I,10	150 PRINT AT 10,I;

El funcionamiento es muy sencillo, porque la figura está definida en una sola línea. Cuando la figura tiene diversas partes, que se colocan en líneas distintas, el movimiento se complica un poco más.

En este primer programa la técnica de dibujo y borrado no tiene ninguna dificultad. La única complicación está en la forma de borrar. Esto se consigue poniendo un espacio en blanco al final del gusano. Cuando tengamos que mover el gusano de una posición n de la pantalla a otra posición n + 1, el espacio en blanco borrará el último carácter de figura anteriormente impresa en la pantalla.

Un ejemplo más complicado, en el cual puede verse cómo el gusano va reptando, es el que aparece en el programa 2.

```
 10 REM ************************************
 20 REM * MOVIMIENTO DE UN GUSANO REPTANTE *
 30 REM * DE IZQUIERDA A DERECHA DE LA PAN *
 40 REM * TALLA.                           *
 50 REM ************************************
 60 REM
 70 REM *** DEFINICION DEL GUSANO ***
 80 REM
 90 LET A$=" ooooo0"
100 LET B$="  oooo0"
110 REM
120 REM *** MOVIMIENTO DEL GUSANO ***
130 REM
140 CLS
150 FOR I=1 TO 25
160     LOCATE 10,I
170     PRINT A$
180     FOR J=1 TO 100
190     NEXT J
200     LOCATE 10,I
210     PRINT B$
220     FOR J=1 TO 50
230     NEXT J
240 NEXT I
250 END
```

Las modificaciones que hay que tener en este segundo programa son las siguientes:

COMMODORE

140 PRINT "<SHIFT-HOME>"
160 LET X=I:LET I=10:GOSUB 9500
200 GOSUB 9500

También hay que incluir la rutina LOCATE PARA COMMODORE que se dio en el primer tomo y que os volvemos a dar en éste en la sección de TRUCOS DE PROGRAMACION.

MSX

160 LOCATE I,10
200 LOCATE I,10

SPECTRUM

160 PRINT AT 10,I;
200 PRINT AT 10,I;

Con este segundo programa ya hemos entrado un poco en las nociones de la animación. Hasta ahora habíamos visto y realizado programas de movimiento, pero en ellos había muy poco de animación. Al trabajar con figuras más complejas la animación resulta más sencilla y más atractiva.

Este programa funciona de una forma muy parecida al anterior, la única diferencia se encuentra en la contracción que hace el gusano cada vez que se mueve una posición a la derecha.

Esta contracción del cuerpo del gusano se consigue, como podéis ver en el listado del programa, imprimiendo encima del gusano de tamaño normal otro gusano más pequeño. El bucle de retardo que hay entre las líneas 220 y 230 hace que esta contracción sea visible. Si este bucle no estuviese, no se podría apreciar dicho movimiento.

Estos dos primeros programas no tienen ninguna dificultad porque, como hemos dicho antes, el movimiento se realiza en una sola línea. El programa 3 simula el movimiento de un ciempiés, que ocupará tres líneas por la pantalla de derecha a izquierda.

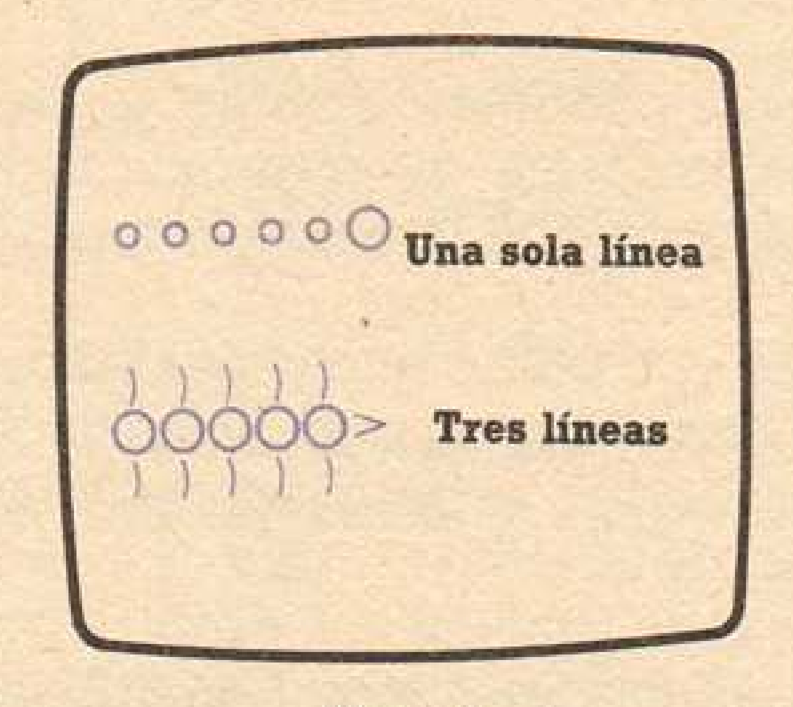


Figura 6.

```
 10 REM *********************************
 20 REM * MOVIMIENTO DE UN CIENPIES DE *
 30 REM * DERECHA A IZQUIERDA DE LA     *
 40 REM * PANTALLA.                     *
 50 REM *********************************
 60 REM
 70 REM *** DEFINICION DEL CIENPIES ***
 80 REM
 90 LET A$=" ((((( "
100 LET B$="<00000 "
110 LET C$=" ((((( "
120 REM
130 REM *** MOVIMIENTO DEL CIENPIES ***
140 REM
150 CLS
160 FOR I=25 TO 1 STEP -1
170     LOCATE 9,I
180     PRINT A$
190     LOCATE 10,I
```

```
200     PRINT B$
210     LOCATE 11,I
220     PRINT C$
230     FOR J=1 TO 100
240     NEXT J
250 NEXT I
260 END
```

Como siempre, para aquellos usuarios de ordenadores distintos del IBM y del Amstrad se incluyen las modificaciones.

COMMODORE

```
150 PRINT "<SHIFT-HOME>"
170 LET X=I:LET Y=9:GOSUB 9500
190 LET Y=10:GOSUB 9500
210 LET Y=11:GOSUB 9500
```

Incluyendo la rutina LOCATE PARA COMMODORE.

MSX

```
170 LOCATE I,9
190 LOCATE I,10
210 LOCATE I,11
```

SPECTRUM

```
170 PRINT AT 9,I;
190 PRINT AT 10,I;
210 PRINT AT 11,I;
```

Como puede apreciarse, en este último programa la forma de imprimir figuras que ocupen más de una línea es imprimirlas línea a línea.

Hay que tener mucho cuidado con el tamaño de estas figuras, pues si las hacemos demasiado grandes, el programa se volverá muy lento y perderá toda su riqueza.

Cuando realices tus propios programas de movimiento con figuras complejas ten en cuenta que, si bien el programa por sí sólo puede ser lo suficientemente veloz como para necesitar un bucle de retardo, cuando estos programas pertenezcan a otro programa más grande, dicho bucle de retardo no a va ser necesario, pues ese tiempo muerto se utilizará para mover otras figuras.

Por fin, y ya para terminar con el movimiento horizontal de figuras complejas, vamos a ver un programa que nos dibuja un ciempiés, pero moviendo las patas.

```
10 REM *********************************
20 REM * MOVIMIENTO DE UN CIENPIES QUE *
30 REM * MUEVE LAS PATAS Y SE DESPLAZA *
40 REM * DE DERECHA A IZQUIERDA        *
50 REM *********************************
60 REM
70 REM *** DEFINICION DEL CIENPIES ***
80 REM
90 REM *** POSICION NUMERO 1 ***
100 REM
110 DIM A$(3):DIM B$(3):DIM C$(3)
120 LET A$(1)=" ((((( "
130 LET B$(1)="<00000 "
140 LET C$(1)=" ((((( "
150 REM
160 REM *** POSICION NUMERO 2 ***
170 REM
190 LET A$(2)=" IIIII "
200 LET B$(2)="=00000 "
210 LET C$(2)=" IIIII "
220 REM
230 REM *** POSICION NUMERO 3 ***
240 REM
260 LET A$(3)=" ))))) "
270 LET B$(3)=">00000 "
280 LET C$(3)=" ))))) "
290 REM
300 REM *** MOVIMIENTO DEL CIENPIES ***
310 CLS
320 REM
330 FOR I=22 TO 1 STEP -3
340    FOR J=3 TO 1 STEP -1
350       LOCATE 9,I+J
360       PRINT A$(J)
370       LOCATE 10,I+J
380       PRINT B$(J)
390       LOCATE 11,I+J
400       PRINT C$(J)
410       FOR K=1 TO 100
420       NEXT K
430    NEXT J
440 NEXT I
450 END
```

Las modificaciones para MSX, COMMODORE y SPECTRUM son las siguientes:

COMMODORE

```
310 PRINT "<SHIFT-HOME>"
350 X=I+J:Y=9:GOSUB 9500
370 Y=10:GOSUB 9500
390 Y=11:GOSUB 9500
```

MSX

```
350 LOCATE I+J,9
370 LOCATE I+J,10
390 LOCATE I+J,11
```

SPECTRUM

```
110 DIM A$(3,7):DIM B$(3,7):DIM C$(3,7)
350 PRINT AT,I+J;
370 PRINT AT 10,I+J;
390 PRINT AT 11,I+J;
```

Este programa, por ser algo más complicado, vamos a explicarlo línea a línea.

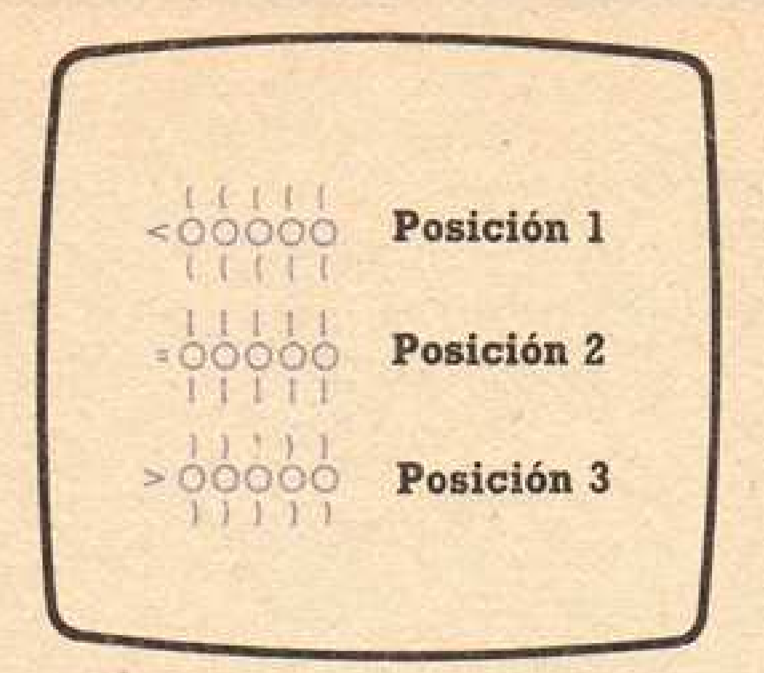


Fig. 7. Estas son las tres posiciones distintas que configuran el movimiento del ciempiés.

Línea 110. Dimensiona tres vectores (variables alfanuméricas de una dimensión) en los que se almacenarán las distintas posiciones del ciempiés.

Líneas 120, 130 y 140. En estas líneas se almacena la primera posición del movimiento del ciempiés. A cada línea de la figura le correponde una variable. Lo hemos hecho de esta manera para que a la hora de realizar el movimiento el programa quede más sencillo.

Líneas 190, 200 y 210. Se almacena la segunda posición del movimiento del ciempiés.

Líneas 260, 270 y 280. Se almacena la tercera y última posición del ciempiés.

Línea 330. Comienza el bucle principal del movimiento. El incremento de dicho bucle está puesto a tres porque tres son los movimientos posibles.

Línea 340. En esta línea empieza un segundo bucle que controlará cada uno de los tres movimientos.

Línea 350. Colocamos el cursor en la primera de las líneas donde tenemos que imprimir la figura.

Línea 360. Imprimimos la primera línea de la figura. En cada vuelta del bucle de la línea 340, como se va variando la variable numérica J, se imprime una primera línea de la figura distinta. Con esto se consigue el efecto de animación.

Línea 370. Se coloca el cursor en la siguiente línea para imprimir la segunda línea de la figura.

Línea 380. Se imprime la segunda línea del ciempiés.

Línea 390. Se coloca el cursor en la tercera línea.

Línea 400. Y se imprime la última línea que configura al ciempiés.

Líneas 410 y 420. Se realiza un bucle de retardo para hacer que el movimiento no sea muy rápido.

Línea 430. Aquí se termina el segundo bucle, el que dice qué figura hay que imprimir.

Línea 440. Se cierra el bucle principal.

Línea 450. Termina el programa.

Si te fijas bien en el programa verás que tiene un pequeño error. Este consiste en que cada movimiento completo del ciempiés no se compone de tres, sino de cuatro movimientos. El orden de las figuras que realizan estos cuatro movimientos son:

Movimiento n.º 1 = Figura 1.
Movimiento n.º 2 = Figura 2.
Movimiento n.º 3 = Figura 3.
Movimiento n.º 4 = Figura 2.

Si te fijas en la figura 8 verás cuáles son estas cuatro posiciones.

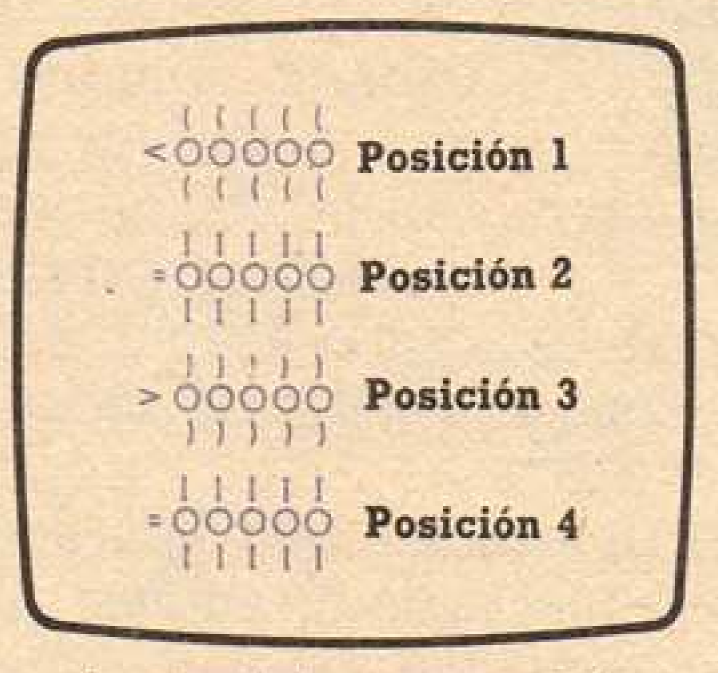


Fig. 8. Estas son las 4 posiciones que tendría que tener el movimiento del ciempiés.

¿Serías capaz de modificar este último programa para que realizase el movimiento de esta nueva forma?

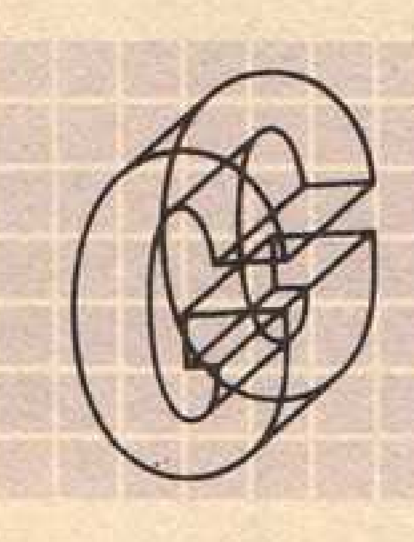
Las modificaciones que tendrías que hacer, línea a línea, son las siguientes:

```
110 DIM A$(4):DIM B$(4): DIM C$(4)
291 REM *** POSICION NUMERO 4 ***
292 REM
293 LET A$(4)=A$(2)
294 LET B$(4)=B$(2)
295 LET C$(4)=C$(2)
296 REM
330 FOR I=21 TO 1 STEP -4
340 FOR J=4 TO 1 STEP -1
```

Los usuarios del SPECTRUM tendrán que cambiar la línea 110 por:

```
110 DIM A$(4,7):DIM B$(4,7):DIM C$(4,7)
```

Para el resto de los ordenadores no hace falta realizar ninguna modificación.



OMO ya hemos visto muchas veces en tomos anteriores, la presentación de los programas de cara al usuario, aunque éste sea uno mismo, es muy importante. Es necesario que nuestros programas sean lo más agradables posible a la vista. Con ello lograremos que el usuario de dicho programa esté más a gusto.

Los programas que aparecen a continuación son una serie de rutinas que nos permitirán imprimir el mensaje PULSA UNA TECLA de una forma más elegante y atractiva de lo que es usual.

Impresión del mensaje 'pulsa una tecla'

Nuestro primer programa imprime la frase PULSA UNA TECLA, de forma que esté parpadeando continuamente.

```
5000 REM ******************************************************
5001 REM *                                                    *
5002 REM *          <<< PULSA UNA TECLA CON PARPADEO >>>      *
5003 REM *                                                    *
5004 REM * VALIDA PARA MSX, AMSTRAD, COMMODORE, IBM Y SPECTRUM *
5005 REM ******************************************************
5006 REM *                                                    *
5007 REM * VARIABLES QUE HAY QUE PASARLE A LA RUTINA          *
5008 REM * -----------------------------------------          *
5009 REM *                                                    *
5010 REM * X = COLUMNA DONDE SE POSICIONARA EL MENSAJE         *
5011 REM * Y = FILA DONDE SE POSICIONARA EL MENSAJE            *
5012 REM *                                                    *
5013 REM * LA SUBRUTINA NO RETORNA NINGUN VALOR               *
5014 REM *                                                    *
5015 REM * VARIABLES USADAS INTERNAMENTE                      *
5016 REM * -----------------------------                      *
5017 REM *                                                    *
5018 REM * I  = CONTADOR DE BUCLE                             *
5019 REM * J  = CONTADOR DE BUCLE                             *
5020 REM * K  = CONTADOR DE BUCLE                             *
5021 REM * A$ = 'PULSA UNA TECLA'                             *
5022 REM * L  = LONGITUD DE A$                                *
```

```
5023 REM *                                                      *
5024 REM ********************************************************
5025 REM
5026 LET A$="PULSA UNA TECLA"
5027 LET L=LEN(A$)
5028 FOR I=1 TO 1 STEP 0
5029    BEEP
5030    FOR J=1 TO 6
5031       LOCATE Y,X
5032       PRINT A$;
5033       IF INKEY$<>"" THEN GOTO 5043
5034       FOR K=1 TO 100
5035       NEXT K
5036       LOCATE Y,X
5037       PRINT SPACE$(L);
5038       IF INKEY$<>"" THEN GOTO 5043
5039       FOR K=1 TO 100
5040       NEXT K
5041    NEXT J
5042 NEXT I
5043 LOCATE Y,X
5044 PRINT SPACE$(L);
5045 RETURN
```

Aunque hay ordenadores que tienen incorporado en el BASIC funciones para hacer que parpadeen los mensajes que se imprimen, con esta rutina puede hacerse que la frase 'PULSA UNA TECLA' esté parpadeando en el lugar que nosotros queramos de la pantalla hasta que el usuario pulse una tecla.



Fig. 1. En el Spectrum podemos hacer que un mensaje parpadee, gracias al comando "FLASH".

El mensaje a imprimir puede ser cualquier otro, mientras no sobrepase los 30 caracteres. Esto es así porque si no la rutina pierde parte de su vistosidad y se hace más lenta.

Este programa puede funcionar sin ningún tipo de problemas en el IBM PC y compatibles y en el AMSTRAD. Para el resto de los ordenadores se proponen las siguientes variaciones:

COMMODORE

Quitar la línea 5029.

```
5031 GOSUB 9500
5033 GET B$:IF B$<>"" THEN GOTO 5043
5035 GOSUB 9500
5036 FOR K=1 TO L:PRINT " ";:NEXT K
5038 GET B$:IF B$<>"" THEN GOTO 5043
5043 GOSUB 9500
5044 FOR K=1 TO L:PRINT " ";:NEXT K
```

También hay que unir el programa con la rutina LOCATE PARA COMMODORE que se dio en el tomo 1 y que vuelve a aparecer en éste.

MSX

```
5031 LOCATE X,Y
5035 LOCATE X,Y
5043 LOCATE X,Y
```

SPECTRUM

```
5029 BEEP 0.3,30
5031 PRINT AT X,Y;
5035 PRINT AT Y,X;
5036 PRINT OVER 1;A$;
5043 PRINT AT Y,X;
5044 PRINT OVER 1;A$;
```

El funcionamiento de la rutina línea a línea es el siguiente:

Línea 5026. Se asigna a la variable alfanumérica A$ el string PULSA UNA TECLA. Esto puede ser cambiado y asignarle otro mensaje distinto, como puede ser PULSA 'F'



Fig. 2. Con el programa 1 podemos hacer que los mensajes que se le dan al usuario para que pulse una tecla sean más agradables.

PARA FINALIZAR. Al ir el mensaje almacenado en una variable, se puede hacer que dicha variable tenga su asignación en otra parte del programa. Con ello conseguimos que esta rutina nos sea útil para imprimir cualquier tipo de mensaje parpadeante en la pantalla.

Línea 5027. Se asigna a la variable numérica L la longitud de la cadena A$.

Línea 5028. Comienza un bucle infinito. Como el incremento (STEP) es igual a cero, este bucle no termina nunca, ya que la variable I nunca llega a tener valor dos. Todos los bucles FORNEXT terminan cuando la variable índice del bucle alcanza un valor superior al límite máximo requerido. Por ejemplo, si tenemos el bucle:

FOR J=1 TO 32:NEXT J

éste terminará cuando la variable J tenga un valor de 33.

Línea 5029. Se hace un ruidito para llamar la atención del usuario.

Línea 5030. Comienza un bucle que dará seis vueltas y en el que se imprimirá el mensaje parpadeando.

Línea 5031. Se coloca el cursor en la posición deseada.

Línea 5032. Se imprime el mensaje. El punto y coma (;) del final le dice al ordenador que después de imprimir el mensaje no cambie de línea. Esto no está hecho así porque este tipo de mensajes se suelen colocar en la última línea de la pantalla.

Línea 5033. Se mira si se ha pulsado alguna tecla del teclado. En caso negativo se continúa con el programa. En el caso de que sí se haya pulsado, se manda a la línea 5043.

Líneas 5034 y 5035. Comienza un bucle en vacío cuya función es ralentizar el parpadeo. La velocidad de éste se puede variar con sólo cambiar el número 100 que aparece después de la palabra reservada TO.

Línea 5036. Se vuelve a colocar el cursor en la posición anterior, pero esta vez para borrar el mensaje.

Línea 5037. Se imprimen tantos espacios en blanco como indique la variable L. L nos dice el número de caracteres del mensaje impreso.

Línea 5038. Igual que la línea 5033. Se mira si se ha pulsado alguna tecla.

Líneas 5039 y 5040. Otro bucle de retardo. Si se varía el tiempo de retardo del primer bucle, también hay que variar el de éste.

Línea 5041. Fin del segundo bucle.

Línea 5042. Fin del primer bucle.

Línea 5043. Coloca el cursor al principio el mensaje.

Línea 5044. Y lo borra.

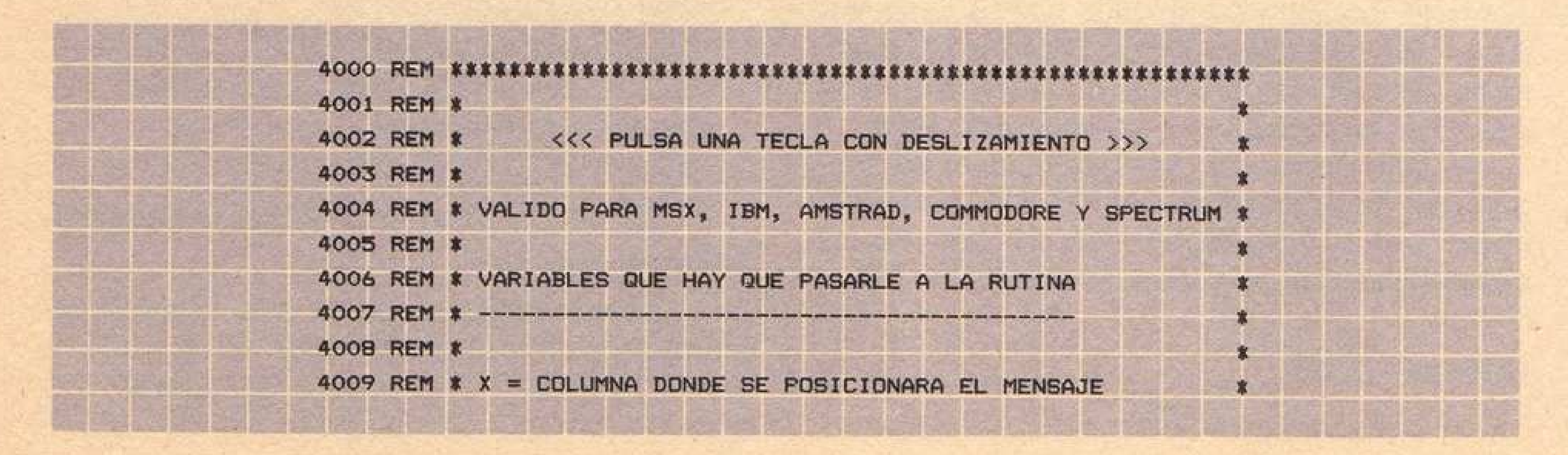
Línea 5045. Vuelve al programa principal.

Estas tres últimas líneas sólo se ejecutan cuando el usuario ha pulsado alguna tecla. Su función es borrar el mensaje PULSA UNA TECLA para que no moleste al seguir ejecutándose el programa.

Como pueden apreciar los usuarios del SPECTRUM, en la línea 5036 aparece lo siguiente:

5036 PRINT OVER 1;A$;

con esto se consigue que el mensaje que está almacenado en A$ se borre de la pantalla.



```
4000 REM *******************************************************
4001 REM *                                                     *
4002 REM *      <<< PULSA UNA TECLA CON DESLIZAMIENTO >>>      *
4003 REM *                                                     *
4004 REM * VALIDO PARA MSX, IBM, AMSTRAD, COMMODORE Y SPECTRUM *
4005 REM *                                                     *
4006 REM * VARIABLES QUE HAY QUE PASARLE A LA RUTINA           *
4007 REM * -----------------------------------------           *
4008 REM *                                                     *
4009 REM * X = COLUMNA DONDE SE POSICIONARA EL MENSAJE         *
```

```
4010 REM * Y = FILA DONDE SE POSICIONARA EL MENSAJE          *
4011 REM *                                                    *
4012 REM * LA SUBRUTINA NO RETORNA NINGUN VALOR               *
4013 REM *                                                    *
4014 REM * VARIABLES USADAS INTERNAMENTE                      *
4015 REM * -----------------------------                      *
4016 REM *                                                    *
4017 REM * I  = CONTADOR DE BUCLE                             *
4018 REM * J  = CONTADOR DE BUCLE                             *
4019 REM * K  = CONTADOR DE BUCLE                             *
4020 REM * A$ = 'PULSA UNA TECLA'                             *
4021 REM * L  = LONGITUD DE A$                                *
4022 REM *                                                    *
4023 REM ******************************************************
4024 REM
4025 LET A$="PULSA UNA TECLA"
4026 LET L=LEN(A$)
4027 LET A$=SPACE$(L)+A$+SPACE$(L)
4028 LOCATE Y-1,X-1
4029 FOR I=1 TO L+2
4030    PRINT CHR$(177);
4031 NEXT I
4032 LOCATE Y,X-1
4033 PRINT CHR$(177);SPACE$(L);CHR$(177)
4034 LOCATE Y+1,X-1
4035 FOR I=1 TO L+2
4036    PRINT CHR$(177);
4037 NEXT I
4038 FOR I=1 TO 1 STEP 0
4039    FOR J=1 TO 2*L+1
4040       LOCATE Y,X
4041       PRINT MID$(A$,J,L)
4042       IF INKEY$<>"" THEN GOTO 4048
4043       FOR K=1 TO 50
4044       NEXT K
4045       IF J=L+1 THEN FOR K=1 TO 700:NEXT K
4046    NEXT J
4047 NEXT I
4048 RETURN
```



Fig. 3. Con el programa 2 el mensaje aparece dentro de la caja.

En el SPECTRUM, la función OVER nos sirve para imprimir en pantalla utilizando la operación XOR.

Si quisiésemos que, en vez de pulsar cualquier tecla se pulse una en especial, como la letra F, habría que cambiar las siguientes dos líneas:

```
5033 IF INKEY$="F" THEN GOTO 5043
5038 IF INKEY$="F" THEN GOTO 5043
```

para que funcione.

El programa que vamos a ver a continuación tiene el mismo fin que el anterior. En este caso el mensaje va a aparecer dentro de una caja que nosotros dibujaremos en la pantalla.

Este programa funciona perfectamente en los ordenadores IBM PC y compatibles. Las líneas que hay que variar para que funcione en los demás ordenadores son las que se dan a continuación.

COMMODORE

```
4027 FOR I=1 TO L:A$=" "+A$+" ":NEXT I
4028 Y=Y-1:X=X-1:GOSUB 9500:Y=Y+1
4030 PRINT CHR$(102);
4032 GOSUB 9500
4033 PRINT CHR$(102);:FOR I=1 TO L:PRINT
" ";:NEXT I: PRINT CHR$(102)
4034 Y=Y+1:GOSUB 9500:Y=Y-1:X=X+1
4036 PRINT CHR$(102);
4040 GOSUB 9500
4042 GET B$:IF B$<>"" THEN GOTO 4048
```

Como siempre, hay que unir este programa a la rutina LOCATE PARA COMMODORE que se dio en el primer tomo y que en éste volvemos a reproducir.

MSX

```
4028 LOCATE X-1,Y-1
4030 PRINT CHR$(207);
4032 LOCATE X-1,Y
4033 PRINT CHR$(207); SPACE$(L)+CHR$(207)
4034 LOCATE X-1,Y+1
4036 PRINT CHR$(207);
4040 LOCATE X,Y
```

SPECTRUM

```
4027 FOR I=1 TO L:LET A$=" "+A$+" ":NEXT I
4028 PRINT AT Y-1,X-1;
4030 PRINT CHR$(143);
4032 PRINT AT Y,X-1;
4033 PRINT CHR$(143);:FOR I=1 TO
L:PRINT" ";:NEXT I: PRINT CHR$(143)
4034 PRINT AT Y+1,X-1;
4036 PRINT CHR$(143);
4040 PRINT AT Y,X;
4041 PRINT A$(J TO J+L)
```

AMSTRAD

Sólo hay que hacer unas pequeñas sustituciones. En todas las líneas donde aparezca:

CHR$(177)

hay que poner:

CHR$(207)

El funcionamiento del programa es muy sencillo, aunque es un poco más complejo que el que hemos visto anteriormente. Lo que hace este programa línea a línea es lo siguiente:

Línea 4025. Asigna a la variable alfanumérica A$ el string PULSA UNA TECLA. Este mensaje puede cambiarse por cualquier otro, como ya vimos anteriormente.

Línea 4026. La variable numérica L almacenará la longitud de A$ en caracteres.

Línea 4027. Se concatena A$ con una serie de blancos tanto por la derecha como por la izquierda, de forma que el mensaje quede justamente en el centro de la variable. El número de blancos que se le suman por la derecha es igual a los que se le suman por la derecha e igual a la longitud de A$.

Línea 4028. Se coloca el cursor una línea más arriba y un carácter más a la izquierda de donde se imprimirá el mensaje.

Línea 4029. Comienza un bucle dentro del cual se dibujará el lado superior de la caja en la que aparecerá el mensaje. Dicha caja tiene un tamaño de L+2x3. Esto es: tiene dos caracteres más de anchura que el mensaje y tiene tres caracteres de altura.

Línea 4030. Se imprime la primera línea de la caja carácter a carácter.

Línea 4031. Aquí termina el bucle.

Línea 4032. Se coloca el cursor en la línea siguiente a la anterior, pero en la misma columna.

Línea 4033. Se imprimen los dos lados laterales (derecho e izquierdo) de la caja. El espacio entre ellos es igual a L.

Línea 4034. Se coloca el cursor una línea más abajo de donde estaba y se sigue conservando la misma columna.

Línea 4035. Este bucle es igual al que aparece entre las líneas 4029 y 4031. Su función es dibujar el lado inferior de la caja.

Línea 4036. Se imprime dicho lado carácter a carácter.

Línea 4037. Se termina el bucle.

Hasta ahora lo que ha hecho el programa ha sido inicializarse. Hemos asignado a A$ y a L los valores que nos interesaban y hemos dibujado una caja en la que se imprimirá el mensaje de pulsar una tecla. A partir de este momento el programa va a imprimir dicho mensaje y a comprobar si se pulsa alguna tecla.

Línea 4038. Comienza un bucle gracias al cual el mensaje irá apareciendo por la derecha de la caja, se desplazará hacia la izquierda, estará un rato a la vista del usuario y volverá a moverse hacia la izquierda hasta desaparecer. En esta línea se comienza con el bucle infinito que ya vimos en el programa anterior, y cuya función es hacer que hasta que no se pulse una tecla el programa no termine.

Línea 4039. Aquí comienza el bucle que

se encargará de imprimir el mensaje de izquierda a derecha, de forma que parezca que se va desplazando.

Línea 4040. Se coloca el cursor en la posición que le hayamos indicado.

Línea 4041. Se imprimen tiras de L caracteres de A$ empezando por el carácter número J. En la primera vuelta del bucle se imprimirán los caracteres que van desde el primero hasta el número L. En la segunda desde el segundo hasta L+1. En la tercera desde el tercero hasta L+2, y así sucesivamente.

Línea 4042. Se pregunta si se ha pulsado una tecla. En caso afirmativo, el programa se termina y se manda su control a la línea 4048. En caso contrario, se continúa con el programa.

Líneas 4043 y 4044. Se realiza un bucle en vacío, como tiempo de retardo, antes de imprimir la siguiente letra del mensaje. El valor de este bucle puede ser variado para que el mensaje aparezca más de prisa o más despacio, según nosotros queramos.

Línea 4045. Se pregunta si todo el mensaje está dentro de la caja. En caso afirmativo se le mantiene un rato en la misma posición para más tarde continuar con el bucle. Después de pasar por esta línea el mensaje irá desapareciendo por la izquierda de la caja.

Línea 4046. Aquí termina el segundo bucle.

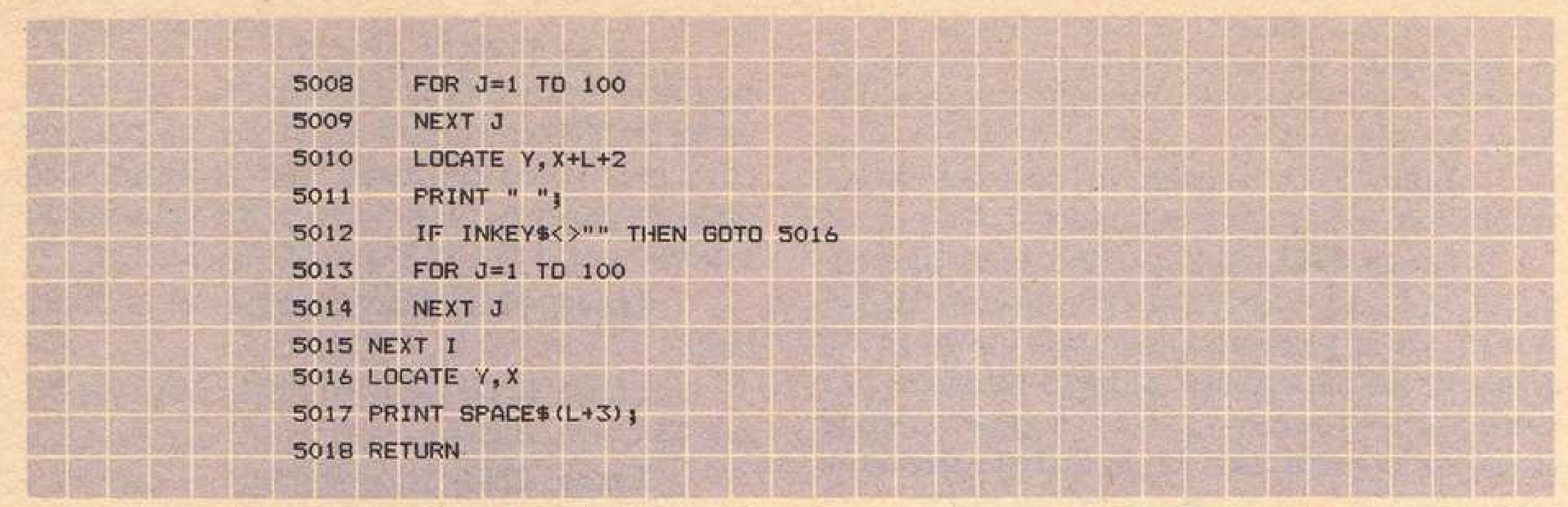
Línea 4047. Aquí termina el primer bucle. El bucle infinito.

Línea 4048. Se devuelve el control al programa principal.

Si os fijáis en el funcionamiento del programa, podréis ver que es muy parecido a uno de los programas de impresión de mensajes que vimos hace algunas semanas. Os recuerdo esto para que veáis que se pueden hacer muchísimas cosas con las rutuinas que aquí vamos proponiendo tomo a tomo.

Otra de las maneras que tenemos de imprimir el mensaje PULSA UNA TECLA lo podemos ver en el programa 3.

```
4000 REM ************************************************************
4010 REM *                                                          *
4020 REM *    <<< PULSA UNA TECLA CON CURSOR PARPADEANTE >>>        *
4030 REM *                                                          *
4040 REM * VALIDO PARA MSX, IBM, AMSTRAD, COMMODORE Y SPECTRUM      *
4050 REM ************************************************************
4060 REM *                                                          *
4070 REM * VARIABLES USADAS POR LA RUTINA                           *
4080 REM * -----------------------------                            *
4090 REM *                                                          *
4100 REM * I   = CONTADOR DE BUCLE                                  *
4110 REM * J   = CONTADOR DE BUCLE                                  *
4120 REM * A$ = 'PULSA UNA TECLA'                                   *
4130 REM * L   = LONGITUD DE A$                                     *
4140 REM *                                                          *
4150 REM * LA RUTINA NO RETORNA NINGUN VALOR                        *
4160 REM *                                                          *
4170 REM * VARIABLES QUE HAY QUE PASARLE A LA RUTINA                *
4180 REM * ------------------------------------------               *
4190 REM *                                                          *
4200 REM * X = COLUMNA DONDE SE POSICIONARA EL MENSAJE              *
4210 REM * Y = FILA DONDE SE POSICIONARA EL MENSAJE                 *
4220 REM *                                                          *
4230 REM ************************************************************
4240 REM
5000 LET A$="PULSA UNA TECLA"
5001 LET L=LEN(A$)
5002 LOCATE Y,X
5003 PRINT A$;
5004 FOR I=1 TO 1 STEP 0
5005    LOCATE Y,X+L+2
5006    PRINT CHR$(177);
5007    IF INKEY$<>"" THEN GOTO 5016
```



```
5008    FOR J=1 TO 100
5009    NEXT J
5010    LOCATE Y,X+L+2
5011    PRINT " ";
5012    IF INKEY$<>"" THEN GOTO 5016
5013    FOR J=1 TO 100
5014    NEXT J
5015 NEXT I
5016 LOCATE Y,X
5017 PRINT SPACE$(L+3);
5018 RETURN
```

Este programa funciona perfectamente en los ordenadores IBM PC y compatibles. Para el resto de los ordenadores hay que introducir los siguientes cambios:

COMMODORE

```
5002 GOSUB 9500
5005 LET X=X+L+2:GOSUB 9500
5006 PRINT CHR$(102);
5007 GET B$:IF B$<>"" THEN GOTO 5016
5010 GOSUB 9500:X=X-L-2
5012 GET B$:IF<>"" THEN GOTO 5016
5016 GOSUB 9500
5017 FOR I=1 TO L+3:PRINT" ";:NEXT I
```

Es necesario unir este programa con la rutina LOCATE PARA COMMODORE que aparece más adelante.

MSX

```
5002 LOCATE X,Y
5005 LOCATE X+L+2,Y
5006 PRINT CHR$(207);
5010 LOCATE X+L+2,Y
5016 LOCATE X,Y
```

AMSTRAD

```
5006 PRINT CHR$(207);
```

SPECTRUM

```
5002 PRINT AT Y,X;
5005 PRINT AT Y,X+L+2;
5006 PRINT CHR$(143);
5010 PRINT AT Y,X+L+2
5016 PRINT AT Y,X;
5017 FOR I=1 TO L+3:PRINT" ";:NEXT I
```

El funcionamiento del programa línea a línea es el siguiente:

Línea 5000. Se asigna a la variable A$ el string PULSA UNA TECLA.

Línea 5001. Se asigna a la variable L el número de caracteres que se almacenan en la variable A$.



Fig. 4. Con el programa 3 podemos ver cómo el cursor parpadea al final de la frase.

Línea 5002. Aquí colocamos el cursor en la posición que le hemos indicado a la rutina al darle valores a las variables X e Y.

Línea 5003. Se imprime el mensaje. El punto y coma (;) del final de la línea sirve para que si el mensaje se coloca en la última línea, la pantalla no haga un scroll.

Línea 5004. En esta línea comienza un bucle dentro del cual se va a encender y apagar un cursor que se encuentra a la derecha del mensaje. Antes de cada parpadeo se preguntará si se ha pulsado alguna tecla.

Línea 5005. Se coloca el cursor dos lugares más a la derecha del final del mensaje.

Línea 5006. Se imprime un cuadrado de color que hará las funciones de un cursor visible.

Línea 5007. Se pregunta si se ha pulsado alguna tecla. En caso afirmativo, se da el control del programa a la línea 5016. En caso contrario, el programa continúa como hasta ahora.

Líneas 5008 y 5009. Se realiza un bucle en vacío que sirve para retardar el parpadeo del cursor.

Línea 5010. Se vuelve a colocar el cursor invisible dos lugares más a la derecha del final del mensaje.

Línea 5011. Se imprime un espacio en blanco para borrar el cuadradito que hacía de cursor. Esta línea, conjugada con la línea número 5006 que imprime un cuadradito, hace que nos parezca que el cursor está parpadeando.

Línea 5012. Se vuelve a preguntar si se ha pulsado alguna tecla.

Líneas 5013 y 5014. Se realiza un nuevo bucle en vacío que actúa como retardo del parpadeo.

Línea 5015. Termina el bucle principal. Este también es un bucle infinito, pues el incremento (STEP) es cero.

Línea 5016. Se coloca el cursor de nuevo al principio del mensaje.

Línea 5017. Se borra dicho mensaje para que no moleste en la posterior ejecución del programa principal.

Línea 5018. Se devuelve el control al programa principal.

Ejercicios resueltos

EJERCICIO N.° 1

Aunque este último programa lo hemos puesto como una rutina que se puede utilizar para imprimir mensajes del tipo:

PULSA UNA TECLA

su mejor aplicación puede encontrarse a la hora de hacer menús por pantalla. En los programas que utilizan menús suele elegirse una cierta opción tecleando, bien por el número que precede a dicha opción, bien la letra por la que empieza dicha opción o bien la letra que se encuentra al principio de dicha opción.

Normalmente, debajo del menú se le suele dar al usuario un mensaje parecido al siguiente:

POR FAVOR,
PULSE LA OPCION QUE LE INTERESE

con el cursor a la derecha de la frase, tal y como aparece en el programa 3.

Para hacer que este último programa nos sirva para introducirlo en la recogida de datos de nuestros menús, sólo hay que realizar algunos cambios. Estos consisten en:

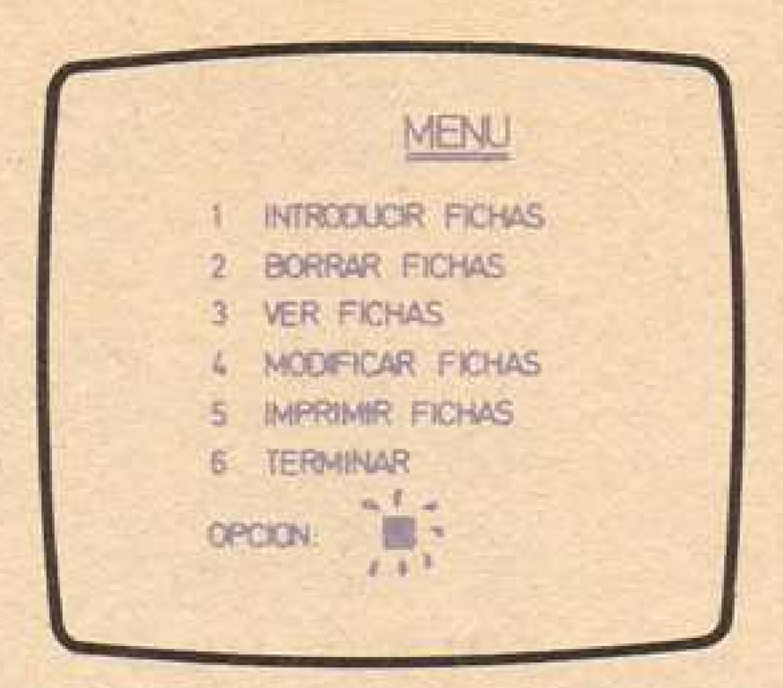


Fig. 5. Forma típica de un menú.

1. Imprimir un mensaje como el visto anteriormente.
2. Aceptar sólo algunas de las teclas. Sólo aquellas que estén reflejadas como opciones en el menú.
3. Sacar por pantalla algún mensaje de error cuando el usuario pulse alguna tecla no permitida.
4. Tiene que aparecer la tecla pulsada en el lugar del cursor una vez elegida la opción.

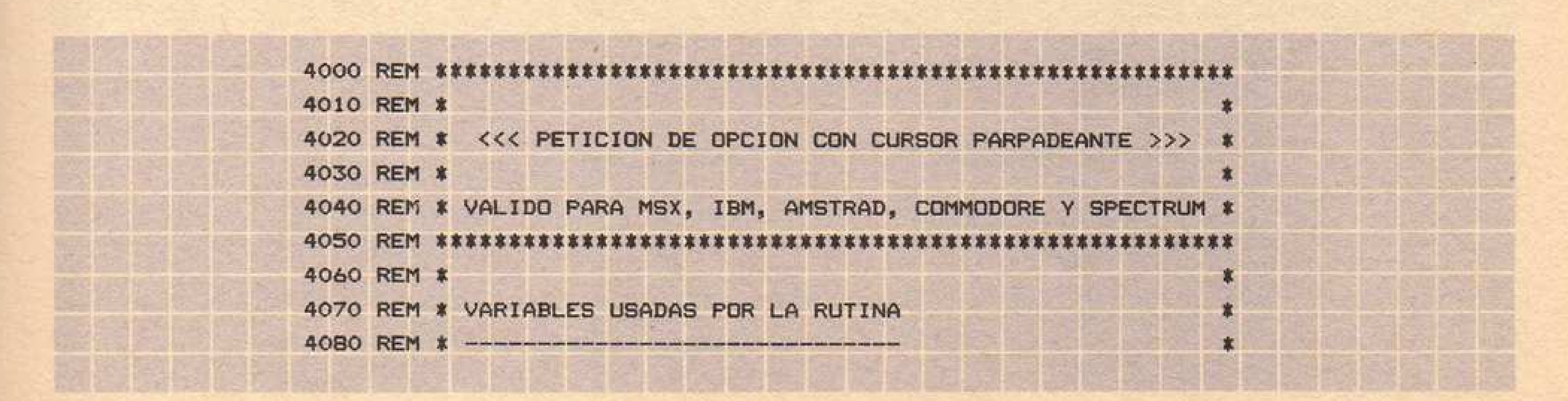
¿Serías capaz de realizar dichos cambios?



Fig. 6. Si el usuario no pulsa la tecla que se le pide, aparece un mensaje de error.

SOLUCION

El programa 4 resuelve este ejercicio de una forma muy sencilla. ¿Se te ocurre otra?



```
4000 REM ******************************************************
4010 REM *                                                    *
4020 REM *  <<< PETICION DE OPCION CON CURSOR PARPADEANTE >>>  *
4030 REM *                                                    *
4040 REM * VALIDO PARA MSX, IBM, AMSTRAD, COMMODORE Y SPECTRUM *
4050 REM ******************************************************
4060 REM *                                                    *
4070 REM * VARIABLES USADAS POR LA RUTINA                     *
4080 REM * -------------------------------                    *
```

```
4090 REM *                                                        *
4100 REM * I  = CONTADOR DE BUCLE                                 *
4110 REM * J  = CONTADOR DE BUCLE                                 *
4120 REM * A$ = 'PULSA UNA TECLA'                                 *
4130 REM * L  = LONGITUD DE A$                                    *
4135 REM * B$ = VARIABLE QUE RECOGE LA ULTIMA TECLA PULSADA       *
4140 REM *                                                        *
4150 REM * LA RUTINA NO RETORNA NINGUN VALOR                      *
4160 REM *                                                        *
4170 REM * VARIABLES QUE HAY QUE PASARLE A LA RUTINA              *
4180 REM * ----------------------------------------               *
4190 REM *                                                        *
4200 REM * X = COLUMNA DONDE SE POSICIONARA EL MENSAJE            *
4210 REM * Y = FILA DONDE SE POSICIONARA EL MENSAJE               *
4220 REM *                                                        *
4230 REM **********************************************************
4240 REM
5000 LET A$="INTRODUZCA OPCION"
5001 LET L=LEN(A$)
5002 LOCATE Y,X
5003 PRINT A$;
5004 FOR I=1 TO 1 STEP 0
5005    LOCATE Y,X+L+2
5006    PRINT CHR$(177);
5007    LET B$=INKEY$
5008    IF B$<>"" AND (B$<"1" OR B$>"7") THEN GOSUB 5023
5009    IF B$<>"" THEN GOSUB 5034:GOTO 5020
5010    FOR J=1 TO 100
5011    NEXT J
5012    LOCATE Y,X+L+2
5013    PRINT " ";
5014    LET B$=INKEY$
5015    IF B$<>"" AND (B$<"1" OR B$>"7") THEN GOSUB 5023
5016    IF B$<>"" THEN GOSUB 5034:GOTO 5020
5017    FOR J=1 TO 100
5018    NEXT J
5019 NEXT I
5020 LOCATE Y,X
5021 PRINT SPACE$(L+3);
5022 RETURN
5023 REM
5024 REM *** ERROR ***
5025 REM
5026 LOCATE Y,X+L+2
5027 PRINT "ERROR ..."
5028 FOR J=1 TO 500
5029 NEXT J
5030 LOCATE Y,X+L+2
5031 PRINT "          "
5032 LET B$=""
5033 RETURN
5034 REM
5035 REM *** IMPRESION DE LA OPCION ELEGIDA ***
5036 REM
5037 LOCATE Y,X+L+2
5038 PRINT B$;
5039 FOR J=1 TO 500
5040 NEXT J
5041 RETURN
```

Las modificaciones a realizar para los ordenadores MSX, COMMODORE y SPECTRUM son muy parecidas a las realizadas en el programa 3, pero, como los números de línea están cambiados, vamos a ponerlas todas de nuevo.

COMMODORE

```
5002 GOSUB 9500
5005 X=X+L+2:GOSUB 9500
5006 PRINT CHR$(102);
5007 GET B$
5012 GOSUB 9500
5101 GET B$
5019 X=X-L-2:NEXT I
5020 GOSUB 9500
5021 FOR J=1 TO L+3:PRINT" ";:NEXT J
5026 GOSUB 9500
5030 GOSUB 9500
5037 GOSUB 9500
```

Como en todos los programas de este libro, hay que unir esta rutina a la que se encuentra un poco más adelante y que se llama RUTINA LOCATE PARA COMMODORE.

MSX

```
5002 LOCATE X,Y
5005 LOCATE X+L+2,Y
5012 LOCATE X+L+2,Y
5020 LOCATE X,Y
5026 LOCATE X+L+2,Y
5030 LOCATE X+L+2,Y
5037 LOCATE X+L+2,Y
```

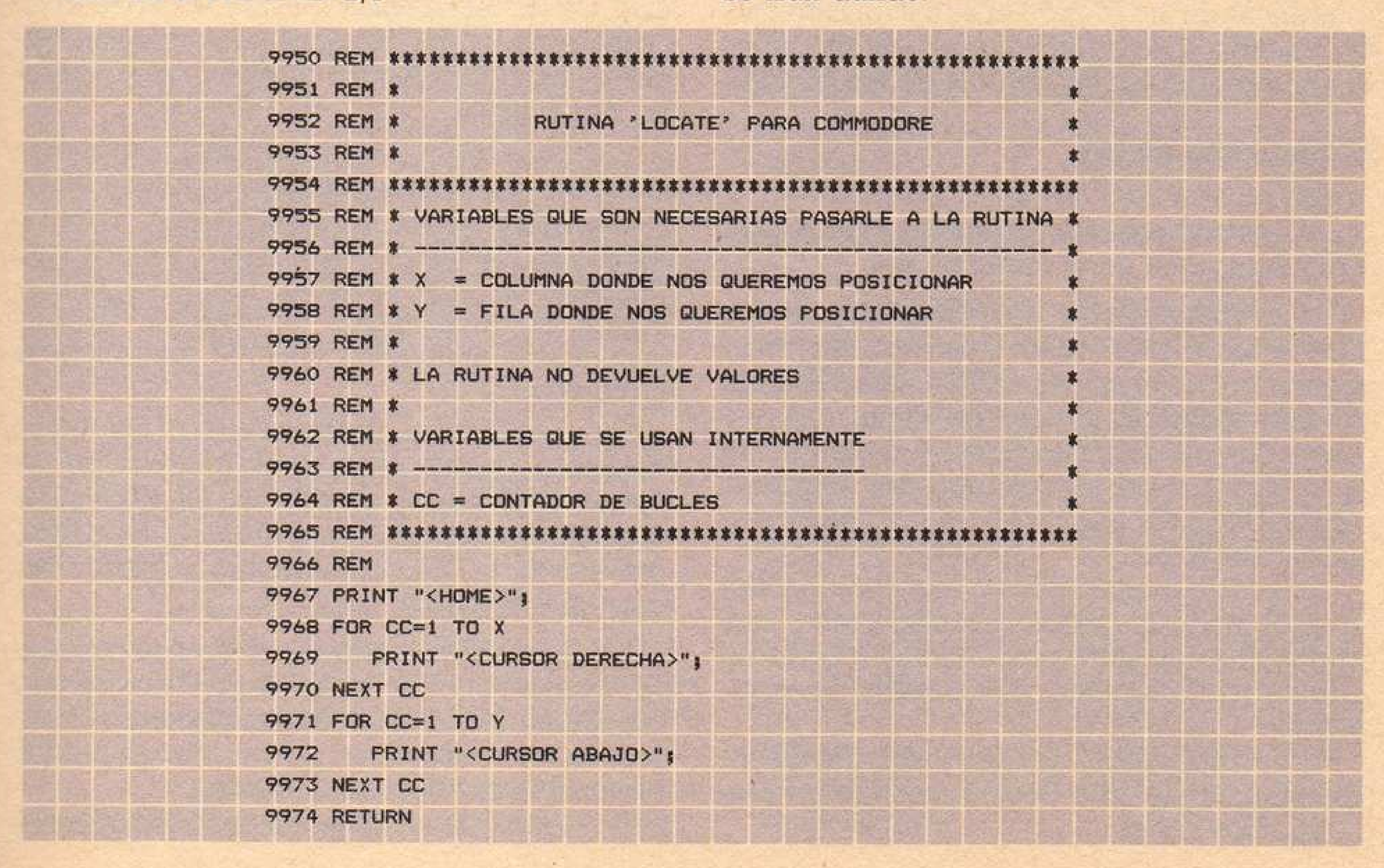
SPECTRUM

```
5002 PRINT AT Y,X;
5005 PRINT AT Y,X+L+2;
5006 PRINT CHR$(143);
5012 PRINT AT Y,X+L+2;
5020 PRINT AT Y,X;
5021 FOR J=1 TO L+3:PRINT" ";:NEXT J
5026 PRINT AT Y,X+L+2;
5030 PRINT AT Y,X+L+2;
5037 PRINT AT Y,X+L+2;
```

Como siempre, el funcionamiento del programa os lo dejo para vosotros. Con ello conseguiréis más velocidad a la hora de programar y se os ocurrirán otras soluciones mejores.

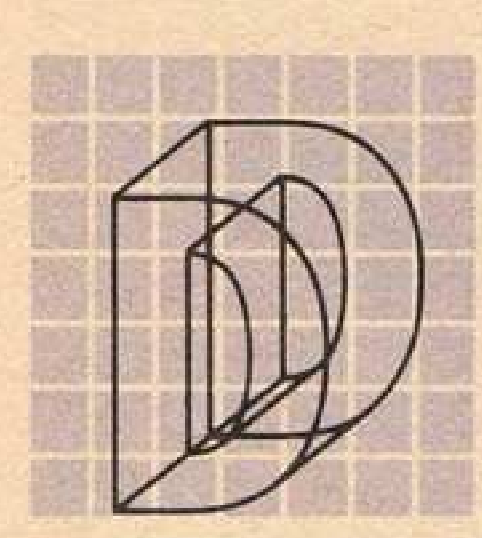
Rutina para simular la sentencia LOCATE en el COMMODORE

Aunque esta rutina apareció en el primer tomo de esta colección, lo volvemos a incluir en éste para aquellas personas que no tengan todavía el tomo primero y para que puedan utilizar todos los programas que aquí se irán dando.



```
9950 REM ****************************************************
9951 REM *                                                  *
9952 REM *          RUTINA 'LOCATE' PARA COMMODORE          *
9953 REM *                                                  *
9954 REM ****************************************************
9955 REM * VARIABLES QUE SON NECESARIAS PASARLE A LA RUTINA *
9956 REM * ------------------------------------------------ *
9957 REM * X  = COLUMNA DONDE NOS QUEREMOS POSICIONAR       *
9958 REM * Y  = FILA DONDE NOS QUEREMOS POSICIONAR          *
9959 REM *                                                  *
9960 REM * LA RUTINA NO DEVUELVE VALORES                    *
9961 REM *                                                  *
9962 REM * VARIABLES QUE SE USAN INTERNAMENTE               *
9963 REM * ---------------------------------                *
9964 REM * CC = CONTADOR DE BUCLES                          *
9965 REM ****************************************************
9966 REM
9967 PRINT "<HOME>";
9968 FOR CC=1 TO X
9969    PRINT "<CURSOR DERECHA>";
9970 NEXT CC
9971 FOR CC=1 TO Y
9972    PRINT "<CURSOR ABAJO>";
9973 NEXT CC
9974 RETURN
```

Conexiones mediante sensores ópticos



E los dispositivos que pueden conectarse a los ordenadores personales hay un conjunto de ellos que utilizan dispositivos sensibles a la luz. Vamos a describir algunos y también aplicaciones interesantes y de fácil realización. Empezaremos describiendo los sensores de señales luminosas y su forma de utilización; después se mostrarán aplicaciones inmediatas de algunos de ellos con posibilidad de conexión a los ordenadores para los cuales se han descrito los conectores de expansión.

Sensores de luz

Los componentes sensibles a la luz y de fácil adquisición en el mercado son:

— Resistencias variables con la luz (LDR = Light Dependent Resistor). Son muy populares y económicas. Son robustas, a la vez que muy sensibles, aunque presentan una respuesta característica no lineal y dependiente de la cantidad de luz recibida. Sin embargo, para variaciones pequeñas de luz tienen respuesta casi lineal.

— Fotodiodos. Los fotones que inciden en ellos generan pares electrón-hueco, que origina una corriente proporcional a la luz incidente. Se emplean polarizados inversamente.

— Diodos PIN. Poseen una capa de semiconductor de bajo dopado, siendo de respuesta muy rápida.

— Fototransistores. Se montan con un fotodiodo que actúa sobre la base de un transistor. Poseen una sensibilidad mucho mayor que los fotodiodos, con un tiempo de respuesta también mayor.

— Fotomultiplicadores. Realizados mediante superficies con sales de fósforo a alta tensión, generan corrientes a partir de luz con un elevado factor de ganancia, pues por cada electrón generado en la primera superficie se generan muchos en cada una de las siguientes. Se emplean solamente cuando la luz recibida es de muy poca intensidad como es el caso de los telescopios o microdensitómetros. Como son también sensibles a fotones en frecuencias no visibles, se emplean para detectar radiaciones.

— CCDs. Charge Coupled Devices = Circuitos Acoplados por Carga, poseen un fotodiodo que carga un condensador. Para su lectura se transfiere el contenido de cada condensador al más próximo, mediante pulsos. Según su organización geométrica, pueden distinguirse: puntuales, lineales, matriciales o circulares.

— Células fotovoltaicas. Generan corriente a partir de la producción de los pares electrón hueco, a partir de los fotones recibidos. Convierten solamente un 10 %, aproximadamente, de la energía recibida en corriente eléctrica.

El circuito mediante el cual se convierte la señal luminosa en señal eléctrica es muy similar en todos ellos, dependiendo los valores de las resistencias y tensiones del rango de luz que sea necesario medir.

En el dibujo se muestra el esquema de

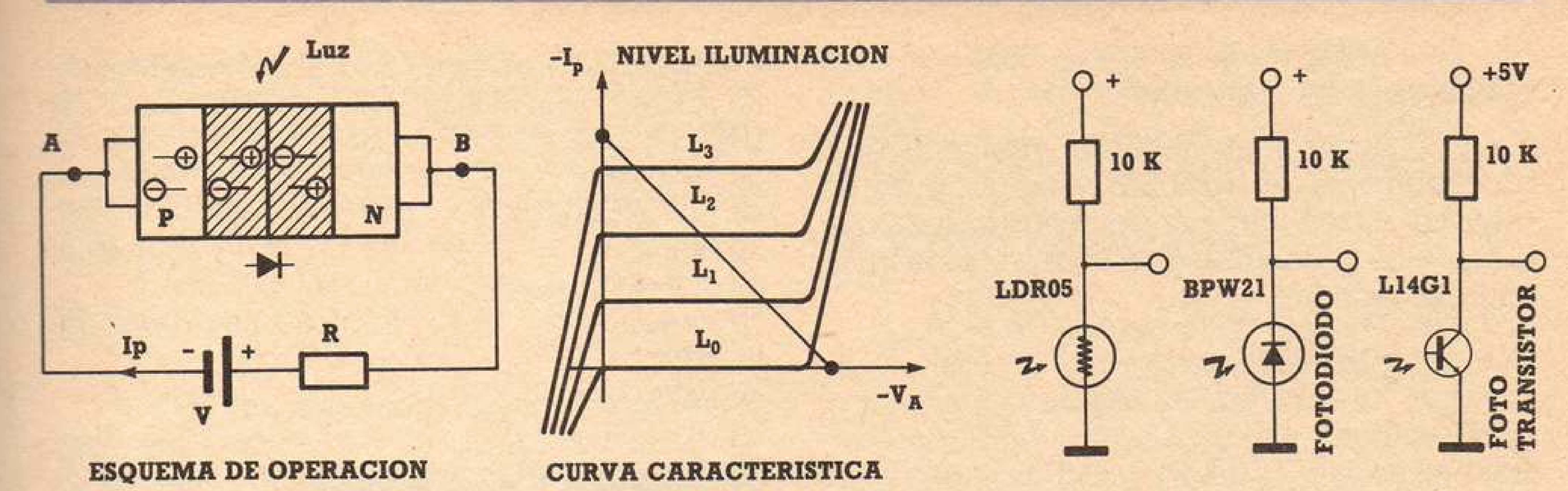


Fig. 1. Fotodiodo.

Fig. 2. Circuitos típicos para sensores de luz.

operación del fotodiodo al ser iluminada su unión PN por la luz. La curva característica muestra la corriente en el fotodiodo para determinados niveles de iluminación, al variar la tensión aplicada a la unión. En condiciones de trabajo normales, la tensión observada entre los bornes del diodo seguiría una recta como la mostrada, pasando por los puntos correspondientes a la tensión de alimentación en el eje horizontal y en la corriente límite, Va/R, en el eje vertical. Es decir, para el esquema de la figura, a mayor nivel de iluminación correspondería menor tensión de salida.

La utilización de la señal depende fundamentalmente de la aplicación, pues en algunos casos es suficiente con determinar si la luz está dentro de un valor especificado, mientras en otros será necesario el empleo de algún tipo de conversor para conocer la magnitud dentro de márgenes precisos. Es conveniente experimentar con la solución que se adopte, variando la resistencia de carga para encontrar el punto de trabajo conveniente. También es recomendable que el punto de trabajo se encuentre, aproximadamente, en el valor medio de la tensión de alimentación si se desea respuesta lineal. Los circuitos indicados no son lineales, presentando su máxima sensibilidad si el punto de trabajo es el indicado. La calibración será necesaria hacerla por comparación con algún fotómetro calibrado, si se desean medidas absolutas, pues la dispersión entre los componentes no permite garantizar unos valores seguros "a priori".

Para el funcionamiento correcto de muchos de los sensores ópticos deberá disponerse de algún filtro de vidrio para eliminar la influencia de los rayos infrarrojos a los que son muy sensibles la mayoría de los fotodiodos y fototransistores.

Los parámetros a tener en cuenta para la selección del dispositivo adecuado son: sensibilidad, respuesta espectral, ángulo de visión, estabilidad con la temperatura, límites admisibles de tensión de trabajo.

Acción de dispositivos desde la pantalla del ordenador

Con los elementos sensores a la luz que hemos descrito, si los colocamos delante de la pantalla de nuestro ordenador personal podemos comunicar órdenes mediante la simple activación de una o varias posiciones en la pantalla. Para ello hemos de seguir los siguientes pasos:

— Deducir mediante un programa sencillo todas las posiciones activas en una línea, por ejemplo, la inferior de la pantalla. Para ello ejecutaremos un programa que presente una secuencia de caracteres como "*" en la línea deseada.

— Seleccionar el número de señales que necesitamos para actuar el equipo exterior.

— Seleccionar la separación. Uno de cada dos puede ser lo apropiado, depende del tamaño del sensor.

— Seleccionar el elemento fotosensible. Un fototransistor, por ejemplo.

— Montar un pequeño armazón para soporte de los fototransistores y colocarlo sobre la pantalla en la línea seleccionada, pegándolo con un adhesivo.

— Añadir los amplificadores para cada fototransistor, hasta conseguir niveles adecuados. Los amplificadores deberán poseer histéresis, pues la pantalla presenta la iluminación de forma parpadeante y aunque posea algo de persistencia, conviene garantizar una conmutación segura.

— Sacar las señales a un conector o cable hasta el equipo a actuar.

El control de equipos mediante este procedimiento es lento pero seguro, pues no requiere introducirse en las interioridades de la máquina y, además, puede servir para cualquier ordenador.

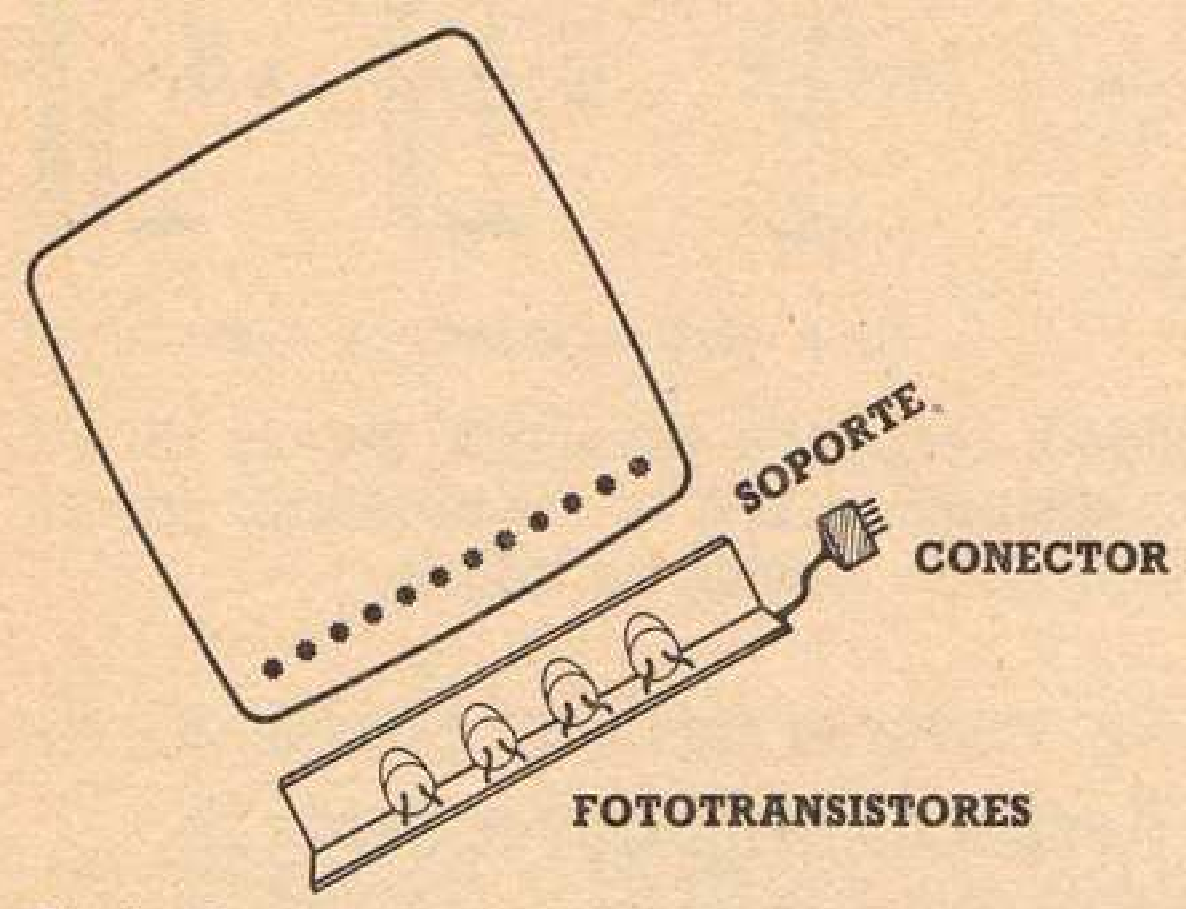


Fig. 3. Disposición de los fototransistores sobre la pantalla.

El circuito de la figura permite calibrar la intensidad luminosa necesaria para la activación. Al iluminarse el fototransistor pasará corriente por él, por lo que habrá tensión alta a la entrada del inversor con histéresis. Para reducir el efecto de parpadeo de la pantalla se incluye el condensador en paralelo con la entrada. La resistencia variable deberá ser ajustada para cada tipo de fototransistor y de pantalla. Para cada elemento sensible deberemos utilizar un circuito como el de la figura, actuando sobre el circuito de control apropiado. El esquema con el diodo LED es solamente para prueba de funcionamiento.

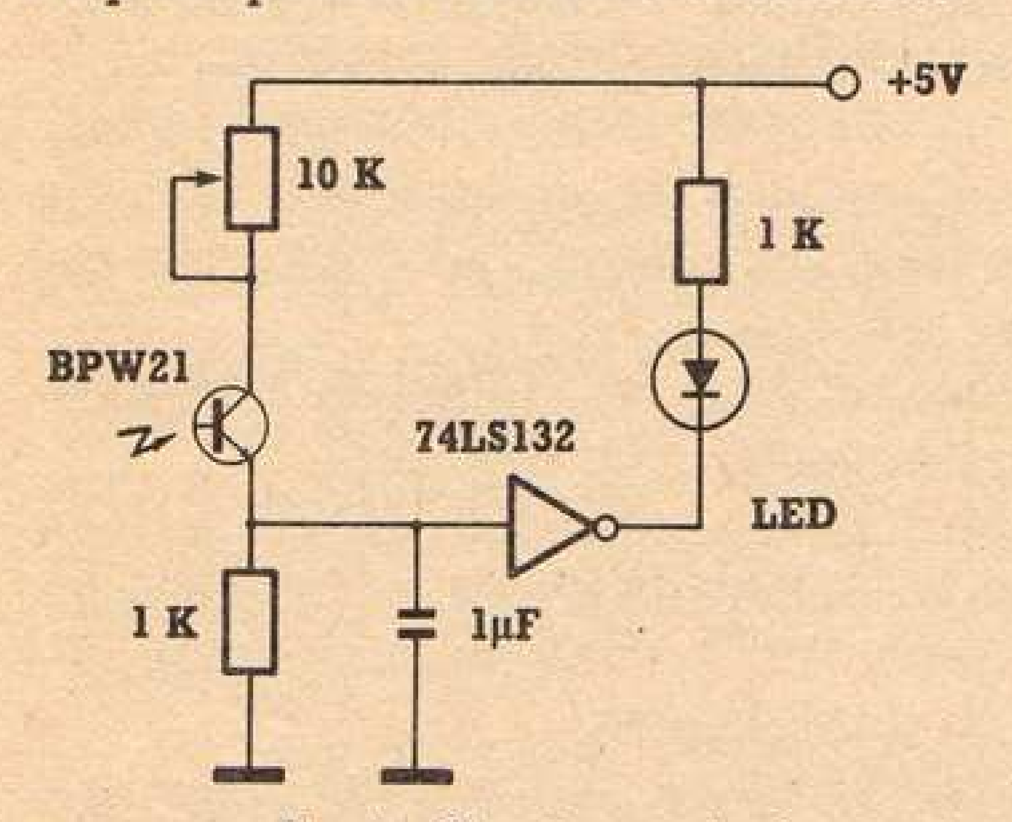


Fig. 4. Circuito para los fototransistores.

El siguiente programa de test sirve para verificación de los sensores para activación desde la pantalla:

```
10 REM PROGRAMA DE ACTIVACION DE DISPOSITIVOS A TRAVES DE LA PANTALLA
20 REM PRODUCE UNA SECUENCIA DE UNOS Y CEROS PARA TEST
30 FOR I=1 TO 1000
40 LOCATE 25,40
50 PRINT "*";
60 LOCATE 25,40
70 PRINT " ";
80 NEXT I
90 END
```

Sensor de luz ambiente

Con un sencillo circuito basado en un sensor luminoso podemos construir un detector de nivel de luz, para poder actuar en consecuencia a través del ordenador. Mediante una fotocélula captamos el nivel de luz en el punto o puntos deseados. Una resistencia variable permite ajustar el nivel que debemos captar para activar un programa o actuar sobre otro dispositivo. Por supuesto, que la acción que desencadene la captación de nivel de luz podría ser una situación de alarma, como, por ejemplo, la aparición de llama.

Si disponemos de varios sensores, calibrados a diferentes niveles, podemos realizar un control proporcional. Si el control fuera necesario hacerlo con mayor resolución, lo apropiado sería disponer un conversor analógico-digital que suministrara el nivel de la señal de manera digital adaptada al margen de medida adecuado.

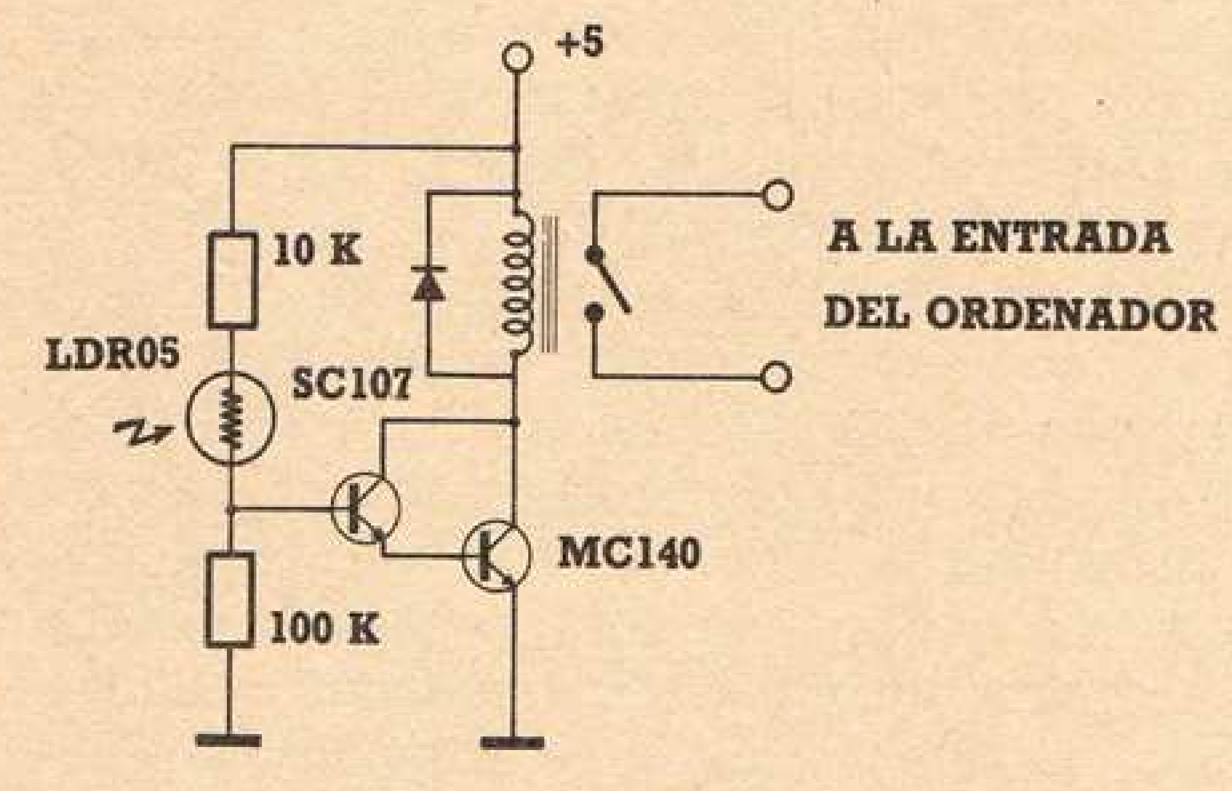


Fig. 5. Circuito de sensor de luz ambiente.

El programa de tratamiento de la lectura debería tener en cuenta la posibilidad de activación esporádica por ruido, por lo que convendrá filtrar la señal digital suministrada y no tomarla en consideración si no ha pasado un determinado período con la señal a nivel estable. El circuito de generación de la señal puede llevar además histéresis para reducir

en lo posible el disparo si las condiciones de entrada fluctúan lentamente entre los valores del punto programado. En el circuito mostrado se ha aislado la comunicación al ordenador mediante un relé actuado mediante el circuito de amplificación del sensor.

Barreras ópticas

Con los elementos descritos podemos realizar barreras ópticas, que pueden utilizarse en muy diferentes aplicaciones. La señal generada en los sensores ópticos, una vez adaptada a los niveles de las entradas del ordenador, pueden conectarse a la tarjeta de ampliación de puertos en cualquiera de los puertos de entrada. Veamos algunos ejemplos concretos:

Barrera de transmisión

Mediante una fuente luminosa, que puede ser no visible si se utilizan LEDS de infrarrojos, se activa un fototransistor. Si se produce una interrupción del rayo luminoso, por el paso de un objeto opaco, la salida del sensor producirá una variación de nivel. Según sea la separación entre la fuente luminosa y el sensor, puede ser necesario utilizar algún medio óptico, como, por ejemplo, un par de lentes, para colimar y concentrar los rayos sobre el fototransistor. El sistema óptico será necesario diseñarlo para cada aplicación específica. Para que los rayos salgan paralelos deberemos colocar la fuente emisora de luz en el foco de la lente y lo mismo del lado receptor. Para que el sistema sólo se active por la luz de la barrera deberemos montar el receptor en un tubo apropiado, de forma solidaria con la lente.

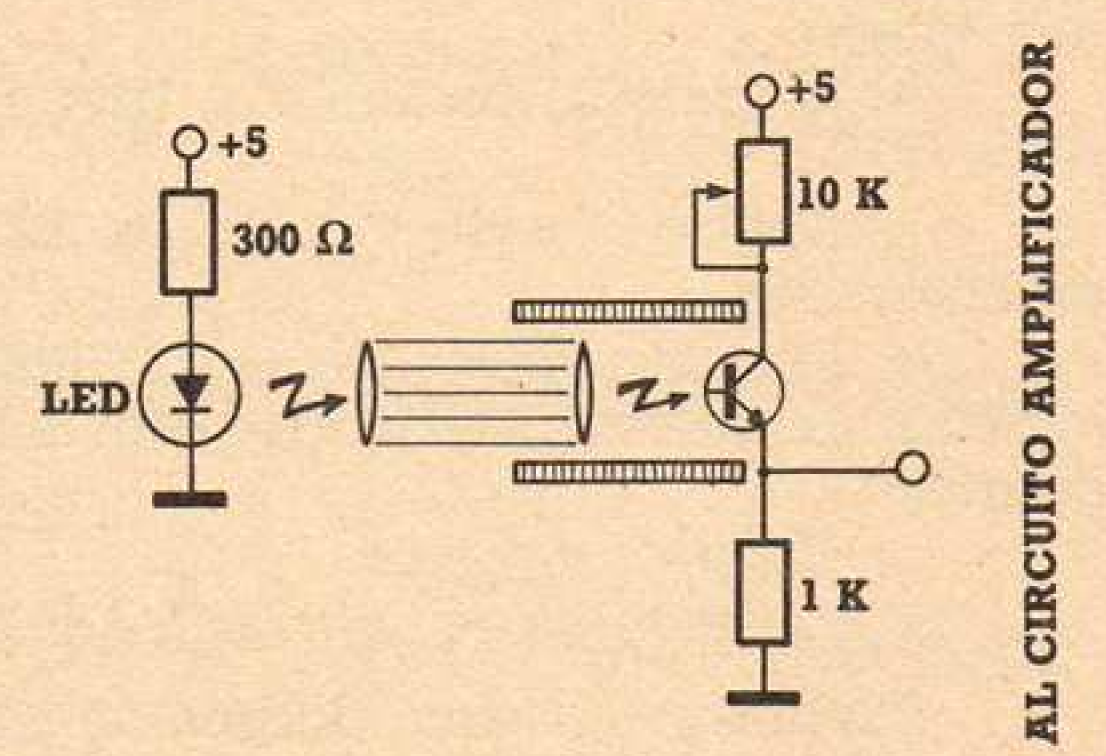


Fig. 6. Esquema de barrera de transmisión.

Este tipo de barreras son aplicables a la detección de paso de piezas o incluso de personas, para medir tiempos entre pasos, por ejemplo. La fuente luminosa puede ser un diodo LED o una lámpara de incandescencia, según la distancia y las condiciones ambientales.

Barrera de reflexión

Para simplificar el montaje de la barrera y eliminar los cables necesarios para la fuente luminosa, puede utilizarse un espejo u otra superficie reflectante, que retorne el rayo luminoso a un punto próximo, donde se colocará el fototransistor del receptor.

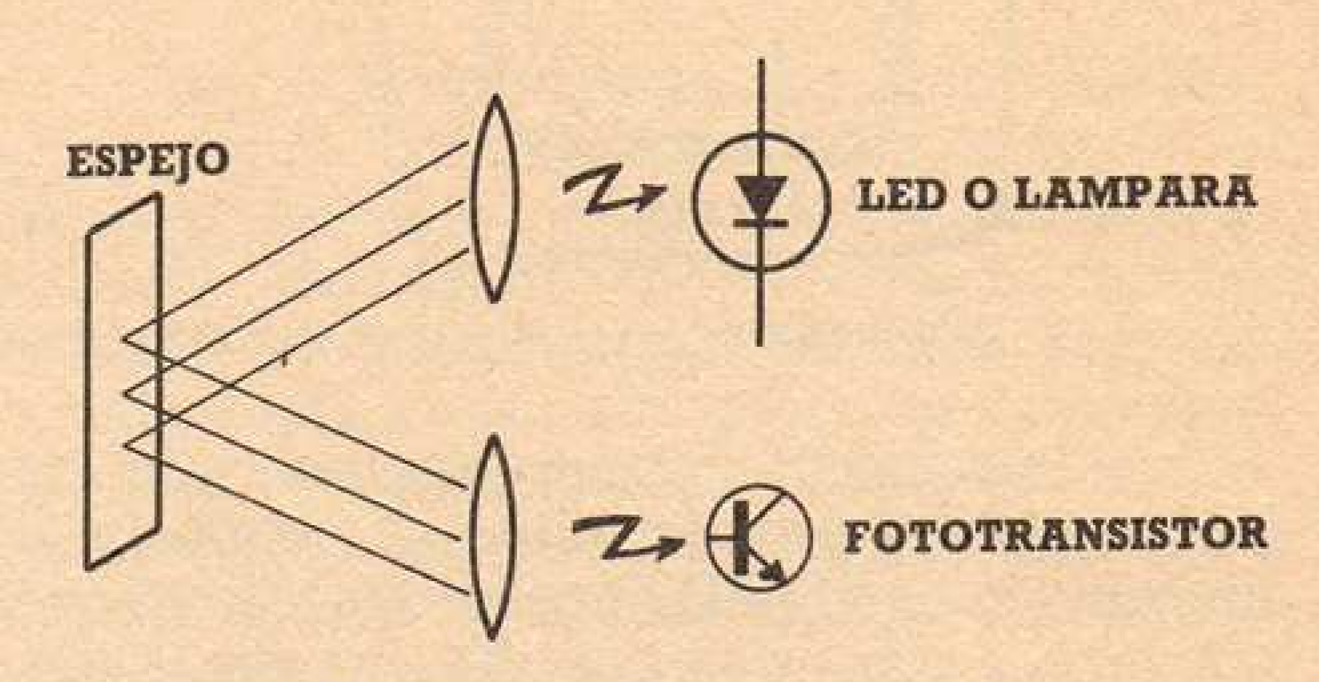


Fig. 7. Esquema de barrera de reflexión.

Para distancias cortas de detección de paso existen parejas de emisor-receptor en un solo bloque. Este tipo de barreras son de aplicación en detección de piezas o de objetos que pasan a distancia fija, como podría ser el caso de vagones de ferrocarril.

Marco para pantalla sensible al tacto

En muchas ocasiones es interesante disponer de un medio de comunicación con el ordenador que permita realizar directamente la selección de las opciones sin más que tocar la pantalla, sin ningún instrumento adicional. Mediante una barrera de rayos infrarrojos puede realizarse este proyecto.

El principio físico consiste simplemente en la creación de franjas de luz infrarroja mediante diodos LEDs y en su captación con fotodiodos o fototransistores. Existe gran variedad de dispositivos que permiten realizar estas barreras y en el diagrama se muestran los circuitos necesarios.

Un aspecto importante a considerar es la resolución necesaria para la aplicación. Para la selección de las opciones de un menú será suficiente con poder presentar como zonas activas el número máximo de opciones posibles, que, en general, no pasarán de 20. Pero

si se desea poder ofrecer las zonas activas en cualquier parte de la pantalla, la resolución efectiva debe ser mayor. Una red de 8 × 8 presenta un número más que suficiente de zonas activas de selección. Los equipos comerciales actualmente disponibles presentan resoluciones hasta de 30 × 40.

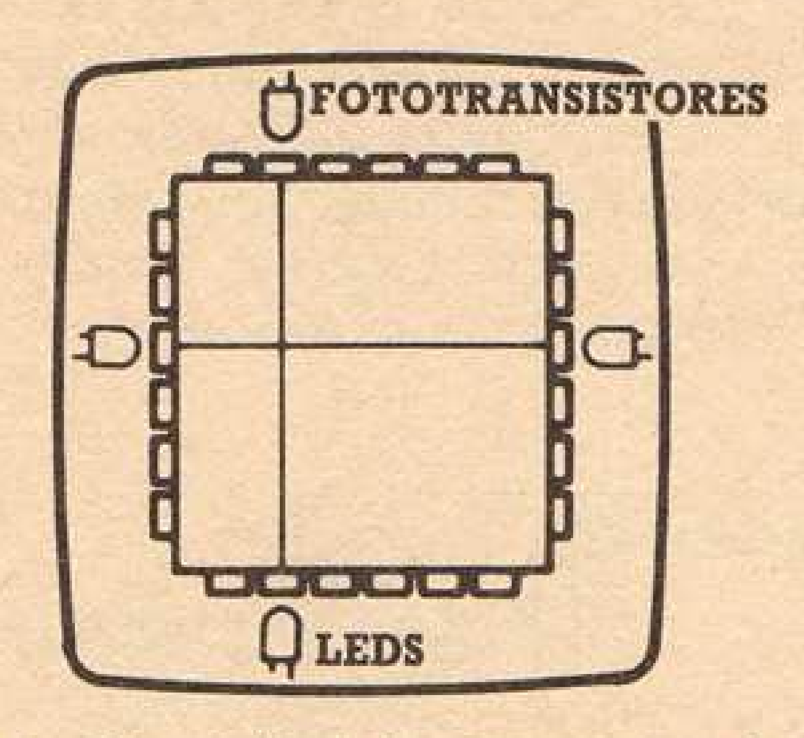


Fig. 8. Disposición de los elementos en el marco.

El programa deberá explorar cada uno de los fototransistores mientras mantiene la iluminación en el LED opuesto. Si detecta que el nivel corresponde a barrera cortada, lo anotará en la tabla de cruces. Si no, anotará un 0. Al terminar el barrido, un pequeño análisis permitirá detectar el punto donde se encuentra el obstáculo y, por tanto, las coordenadas del punto seleccionado. La exploración del LED emisor y del fototransistor receptor puede efectuarse de manera simultánea, o bien pueden alimentarse continuamente todos los LEDs y explorar solamente los fototransistores. El segundo procedimiento tiene la ventaja de la simplicidad, pero puede ocurrir que, por error de alineamiento, un fototransistor reciba iluminación de un LED próximo, con lo que no se detectará la ruptura de la barrera. El primer procedimiento requiere algo más de circuitería, pero no presenta problema de alineamiento, incluso en el caso de poner los LEDs muy juntos. Tanto los LEDs como los fotodetectores se fabrican con una lente que les hace muy direccionales, presentando ángulos de visión activa de 5 a 10 grados.

Los circuitos de alimentación de los LEDs y de recepción con fototransistores son los típicos para las barreras ópticas que hemos visto. Para los componentes con lente incorporada se muestran en la figura.

Los marcos sensibles al tacto suelen conectarse como dispositivo asíncrono serie, generando el adaptador de soporte una secuencia de señales correspondiente a las coordenadas de la celda activada.

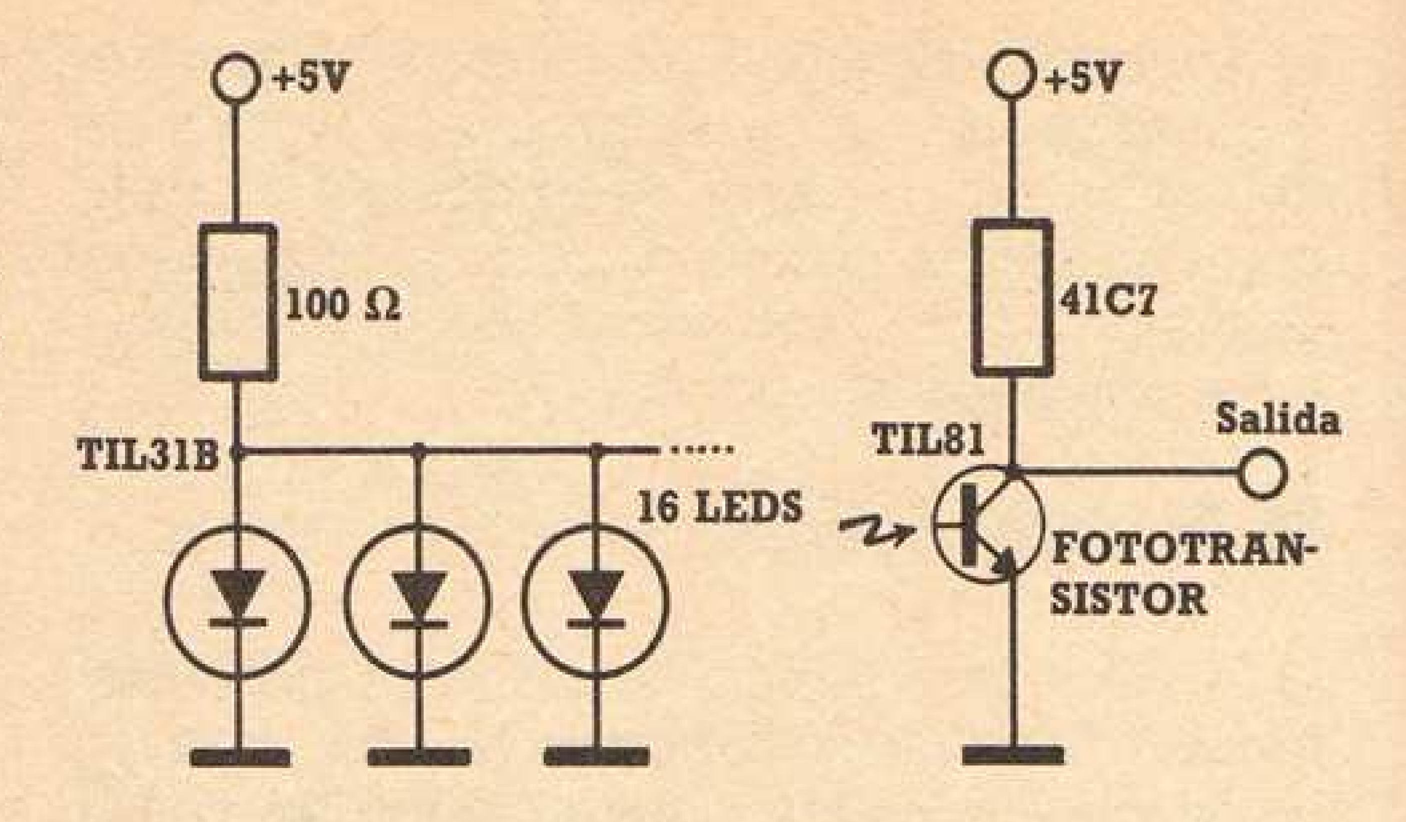


Fig. 9. Circuitos de emisión y recepción.

Es necesario mencionar algunos de los problemas que presenta un sistema de comunicación con la máquina a través del contacto sobre la pantalla: puede resultar incómodo tener que levantar los brazos continuamente para seleccionar la opción, la luz ambiental puede ocasionar activaciones erróneas, el polvo puede obstruir los orificios de paso de luz y además la pantalla acaba siempre con las marcas de los dedos. Sin embargo, puede tener futuro en aplicaciones donde un mismo usuario utiliza con poca frecuencia el sistema o donde pueda protegerse de la suciedad o el deterioro voluntario.

Ejemplo práctico de aplicación

Para desarrollar el esquema presentado necesitamos las siguientes unidades funcionales a construir:

— Marco de soporte adaptado a la pantalla. La superficie de la pantalla suele ser curvada, por lo que se adaptará el soporte para que los rayos de los LEDs sobrepasen el punto más alto.

— 16 parejas de LEDs y fototransistores.

— Cuatro tiras de soporte de LEDs y fototransistores.

— Tarjeta soporte de la circuitería de interfaz, además de la tarjeta de ampliación de puertos.

— Cables de señales.

— Conectores de unión entre unidades.

Pueden plantearse diferentes situaciones según el número de puntos en los que se detecta corte de la barrera. El programa deberá decidir la opción más razonable, calcular un punto medio, por ejemplo, el centro de

gravedad de la zona activada o dar una indicación de error para que realice una nueva selección. El programa que se muestra da como resultado un carácter en pantalla de los puntos activados. Este programa es aplicable al IBM-PC. Para adaptarlo a SPECTRUM o AMSTRAD se deberán cambiar las direcciones de los puertos, que deberán ajustarse a las que produzcan la señal SELR0 y SELR1, según se indicó en el fascículo 1. También se cambiará la sentencia de entrada INP por IN. Para COMMODORE, además deberá sutituirse la sentencia INP por el correspondiente PEEK.

```
10 REM MARCO
20 REM PROGRAMA DE LECTURA DE MARCO DE REJILLA INFRARROJOS
30 REM PARA CUALQUIER OP CON TARJETA DE EXPANSION DE PUERTOS
35 REM IBM-PC, SPECTRUM, AMSTRAD. COMMODORE USAR CAMBIOS
40 REM LEE ESTADO, SI ACTIVO, LEE COORDENADAS X E Y
50 REM LAS ENTRADAS DE X E Y DEBERAN ESTAR CABLEADAS
60 REM EN LOS PUERTOS $H201 Y &H203 DE LA TARJETA DE EXPANSION
70 PUERTOX=&H201: PUERTOY=&H203
80 V$="12345678"
90 CLS: KEY OFF
100 A=INP(PUERTOX): B=INP(PUERTOY)
110 LOCATE 24,1: PRINT HEX$(A);HEX$(B); : REM MUESTRA ESTADO
120 IF (A<>255) OR (B<>255) THEN 130 ELSE 100
130 X=INP(PUERTOX): Y=INP(PUERTOY)
140 X=24: Y=48: REM VALORES SIMULADOS, PARA PRUEBA SIN MARCO
150 GOSUB 200: REM MUESTRA VECTORES VX, VY
160 GOSUB 310: REM PRESENTA MATRIZ
170 LOCATE 15,1: PRINT "PULSE PARA SEGUIR"
180 INPUT A$
190 CLS: GOTO 90
200 REM MUESTRA EN PANTALLA LOS VECTORES
210 LINEA=1
220 FOR I=1 TO 8
230 VX(I)=X AND LINEA: VY(I)=Y AND LINEA
240 LINEA=LINEA*2
250 NEXT I
260 PRINT: PRINT "VX=";
270 FOR I=1 TO 8: PRINT VX(I);: NEXT I
280 PRINT " VY=";
290 FOR I=1 TO 8: PRINT VY(I);: NEXT I: PRINT
300 RETURN
310 LOCATE 1,1: REM MUESTRA LA MATRIZ
320 PRINT " 12345678";
330 FOR I=3 TO 10:LOCATE I,1:PRINT MID$(V$,I-2,1);: NEXT I
340 FOR I=1 TO 8
350 FOR J=1 TO 8
360 IF VX(I)<>0 AND VY(J)<>0 THEN LOCATE 2+I,2+J:PRINT "*";
370 NEXT J
380 NEXT I
390 RETURN
```

La rutina de exploración puede activarse de forma continua, por temporización o bien por detección de que hay algún sensor activado. Mediante el circuito que se indica, podría generarse interrupción cada vez que hubiera algún sensor activado, comenzándose entonces la exploración para detectar cuál o cuáles fueron los que lo causaron. Requiere una entrada adicional para indicar el estado o para producir la interrupción.

Para obtener las condiciones óptimas de funcionamiento conviene alinear las zonas sensibles con posiciones de filas y columnas de la pantalla. Para ello se diseñará una pantalla similar al menú más complejo que se pueda utilizar en la práctica y se harán coincidir las posiciones de las zonas sensibles con las filas y columnas de los puntos de selección del menú. Sabiendo estas coordenadas, los programas que usen esta interfaz táctil utilizarán para indicar los puntos a contactar solamente los centros de las zonas sensibles. Todas las pantallas pueden ajustar su imagen en altura y anchura, por lo que podrá hacerse un ajuste fino para centrado de las zonas sensibles, si se cambia de pantalla.

Lista de componentes electrónicos

— 16 LEDs infrarrojos, TIL31B.

— 16 fototransistores, TIL81.

— Multiplexor, realizado con 74LS05, con salidas conectadas en OR, cableado si se emplea entrada por interrupción.

— Decodificador para la tarjeta de ampliación de puertos, 74LS138.

— Amplificador de conexión al bus, 74LS244.

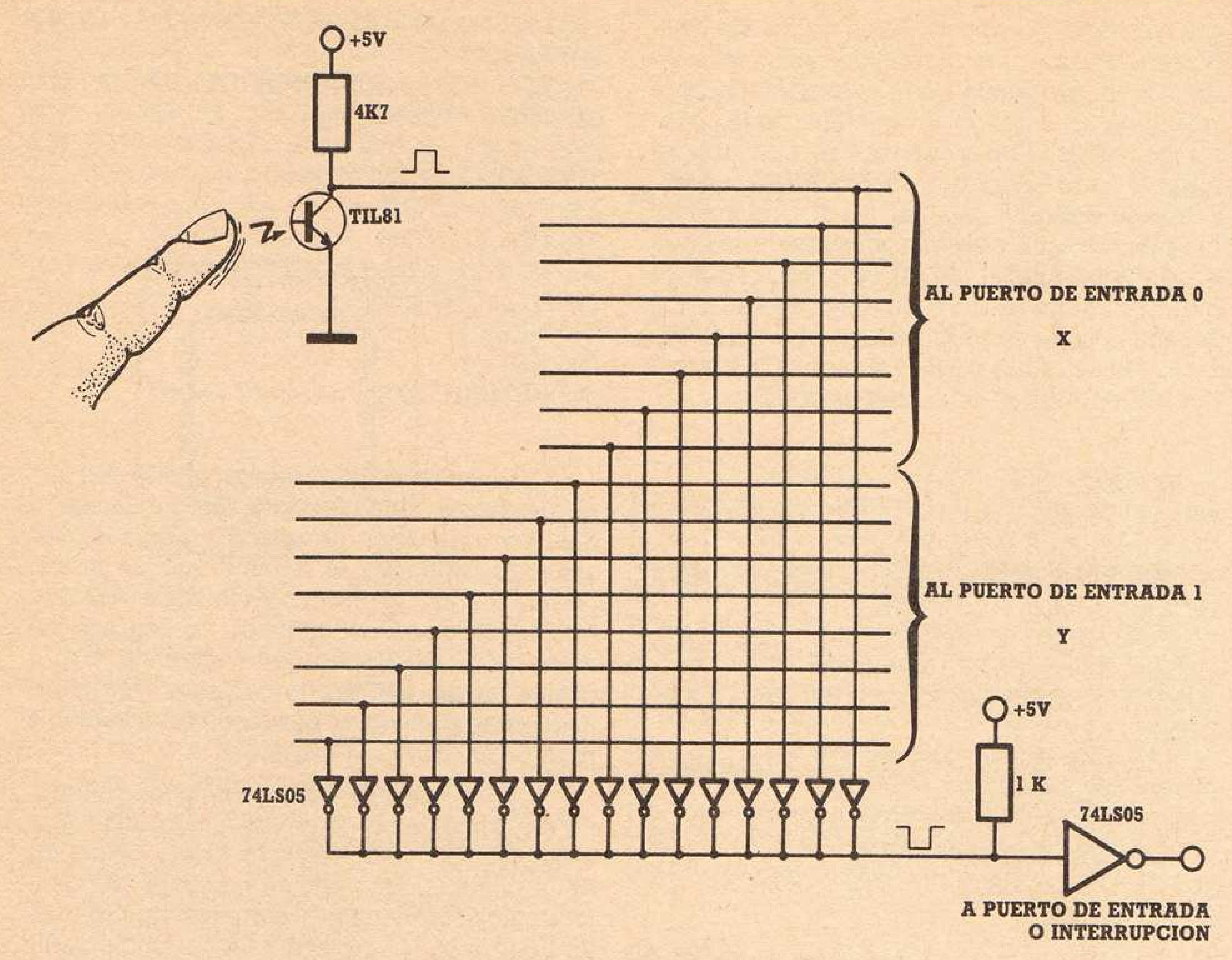


Fig. 10. Circuito para interrupción o indicación de estado.

— Resistencias de 4K7, para los fototransistores.

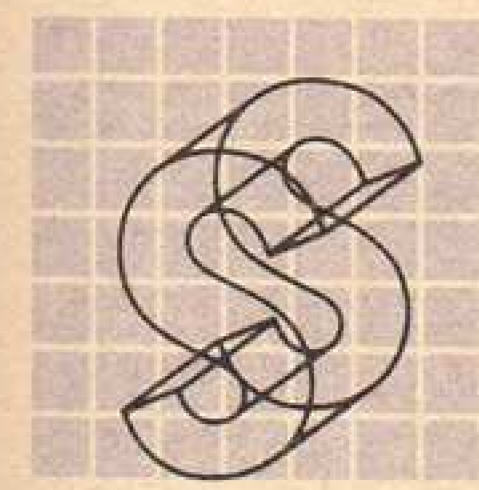
Otras aplicaciones

Hemos visto diversas aplicaciones de los sensores ópticos conectados a un ordenador personal como elementos de entrada que permiten la relación con el mundo luminoso exterior. Solamente hemos contemplado aquellos casos que permiten discriminar en el sensor entre dos niveles diferentes. Existen otras muchas aplicaciones en las cuales es necesario medir el nivel luminoso con precisión o una magnitud que pueda transformarse en nivel luminoso. Para estos casos deberemos disponer de un circuito que convierta el nivel en una cantidad manipulable por el ordenador, esto es, un conversor de magnitudes analógicas en digitales. Para otras aplicaciones, además, hemos de ser capaces de captar una larga secuencia de valores que se producen en un breve período de tiempo y con unas condiciones de temporización determinadas. Para estos casos diseñaremos los circuitos de apoyo necesarios para "congelar" las señales y después darles el tratamiento oportuno.

En los sucesivos capítulos veremos la forma de diseñar y montar estos diferentes circuitos para tener acceso a otros tipos más interesantes de señales.

NATURALEZA Y TECNOLOGIA

Programa de conversión de temperaturas



IEMPRE que tocas un cuerpo, tu sentido del tacto te permite hacer una estimación aproximada de su temperatura; sin embargo, no podemos hacerlo de una forma cuantitativa.

Además debemos tener en cuenta que los efectos producidos por la conductividad de los cuerpos dan lugar a confusión en la apreciación de la temperatura mediante sensaciones fisiológicas. Por ello, el metal nos parece más frío que la madera, sin que ello signifique que están a distinta temperatura.

Para poder medir con aparatos la temperatura de un cuerpo, se recurre al hecho de que las variaciones de temperatura van acompañadas de variaciones en alguna propiedad del cuerpo. Por ejemplo, los cuerpos aumentan de volumen al ser calentados y se contraen al ser enfriados.

Este es el caso de los termómetros de dilatación, en los que una columna líquida varía su longitud en función de la temperatura, asignando de esta forma a cada longitud un valor numérico, obteniéndose la cuantificación deseada.

Y así llegamos a las escalas de temperatura, que van a depender del valor numérico asignado a cada longitud de la columna de líquido.

Las tres escalas que debes conocer son:

- CELSIUS (o escala centígrada).
- KELVIN (o escala absoluta).
- FARENHEIT.

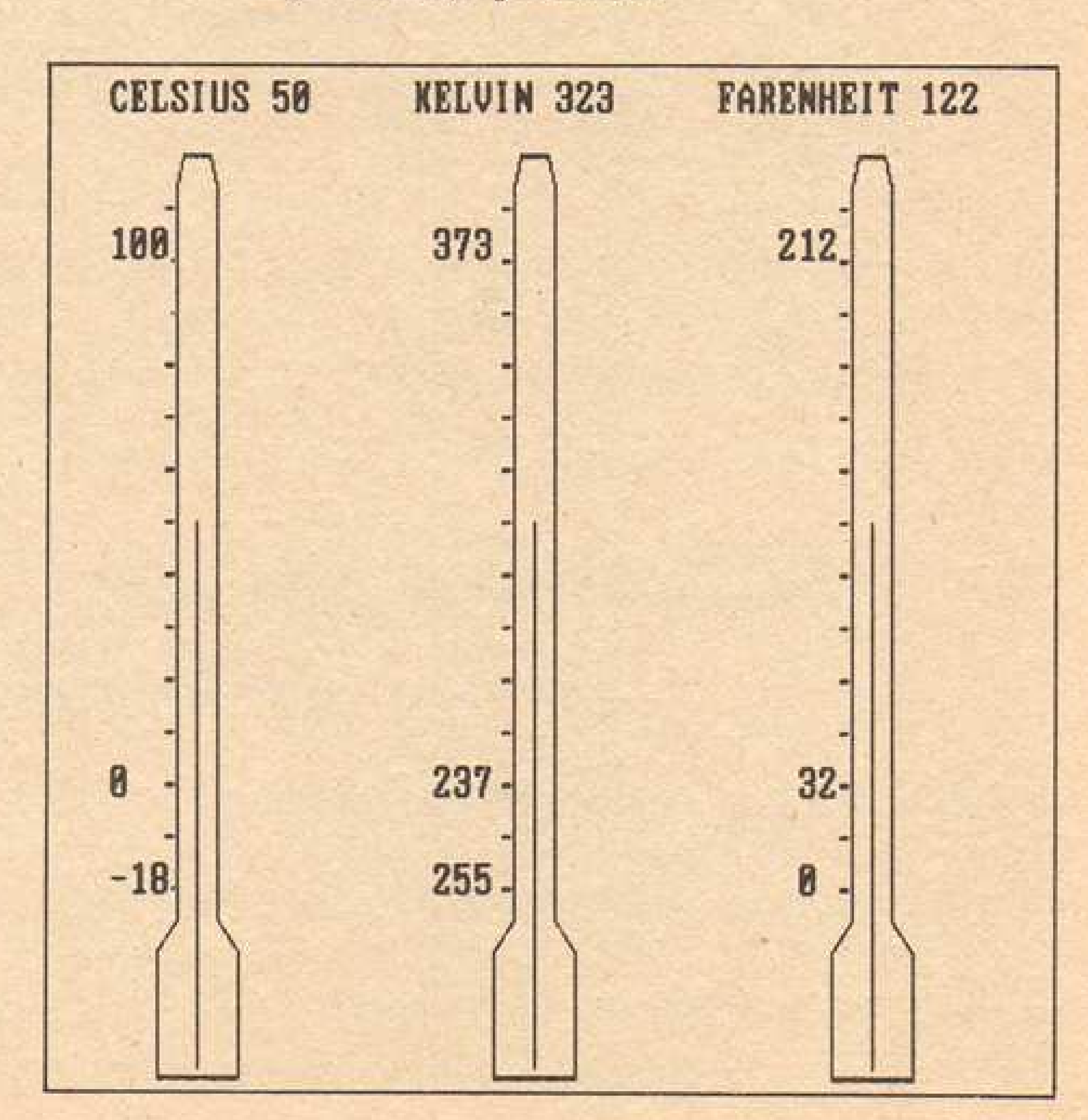
La escala Celsius se caracteriza por tener 0° C en el punto de fusión del hielo y 100° C en el punto de ebullición del agua.

Kelvin se basa en las propiedades de los gases ideales. Las temperaturas del hielo fundente y el agua en ebullición corresponden a 273,15° K y 373,15° K respectivamente.

Se define esta escala a partir del llamado punto triple del agua, que es el punto en el que coinciden en equilibrio los tres estados (sólido, líquido y gaseoso) del agua.

Con ello el Kelvin, unidad de temperatura termodinámica del sistema internacional, se define como la fracción 1/273,16 de la temperatura termodinámica del punto triple del agua.

A la temperatura de 0° K se le llama cero absoluto, ya que a esa temperatura las moléculas cesan de moverse; por tanto, es la menor temperatura posible.



```
1 REM ********************************************
2 REM ** PROGRAMA DE CONVERSION DE TEMPERATURAS **
3 REM ********************************************
10 CLS
20 SCREEN 2
30 PRINT "QUE OPCION DESEAS"
40 PRINT "1/ PASAR DE CELSIUS A KELVIN Y FARENHEIT"
50 PRINT "2/ PASAR DE KELVIN A CELSIUS Y FARENHEIT"
60 PRINT "3/ PASAR DE FARENHEIT A CELSIUS Y KELVIN"
65 PRINT :PRINT :PRINT
70 INPUT R
80 IF R=1  THEN GOSUB 300
90 IF R=2 THEN GOSUB 400
100 IF R=3 THEN GOSUB 500
110 END
300 REM *************************************
301 REM ** SUBRUTINA CONVERSION DE CELSIUS **
302 REM *************************************
305 INPUT " QUE TEMPERATURA TIENES EN GRADOS CELSIUS";TC
310 CLS
320 LET TK=TC+273
330 LET TF=INT(1.8*TC+32)
340 PRINT "CELSIUS";TC;"         KELVIN";TK;"         FARENHEIT";TF
350 GOSUB 1000
360 GOSUB 2000
370 GOSUB 3000
380 RETURN
400 REM ************************************
401 REM ** SUBRUTINA CONVERSION DE KELVIN **
402 REM ************************************
405 INPUT " QUE TEMPERATURA TIENES EN GRADOS KELVIN";TK
410 CLS
420 LET TC=TK-273
430 LET TF=INT(1.8*TC+32)
440 PRINT "CELSIUS";TC;"     KELVIN";TK;"     FARENHEIT";TF
450 GOSUB 1000
460 GOSUB 2000
470 GOSUB 3000
480 RETURN
500 REM ***************************************
501 REM ** SUBRUTINA CONVERSION DE FARENHEIT **
502 REM ***************************************
505 INPUT " QUE TEMPERATURA TIENES EN GRADOS FARENHEIT";TF
510 CLS
520 LET TC=(TF-32)/1.8
530 LET TK=TC+273
540 PRINT "CELSIUS";TC;"     KELVIN";TK;"     FARENHEIT";TF
550 GOSUB 1000
560 GOSUB 2000
570 GOSUB 3000
580 RETURN
1000 REM ******************************************
1001 REM ** SUBRUTINA DE DIBUJO DE TERMOMETROS **
1002 REM ******************************************
1005 LET Y=190
1010 LET X=20
1020 FOR J=1 TO 3
1030 PSET (X,Y)
1040 FOR I= 1 TO 10
1050 READ A,B
1060 LET A=A*2: LET B=B*2
1070 LET A$=STR$(A):LET B=B*(-1):LET B$=STR$(B)
1080 LET DIB$="M"+"+"+A$+","+"+"+B$
1090 DRAW DIB$
1100 NEXT I
1110 LET X=X+150
1120 NEXT J
1130 RETURN
2000 REM **********************************
2001 REM ** SUBRUTINA DE DIBUJO ESCALAS **
2002 REM **********************************
2005 LET A=27
2010 FOR I=1 TO 3
2020 LET B=Y-36
2030 FOR J=1 TO 14
2040 PSET (A,B)
2050 DRAW "M-3,+0"
2060 LET B=B-10
2070 NEXT J
2080 LET A=A+150
2090 NEXT I
```

```
2100 FOR I=1 TO 9
2110 READ A,B,C$
2120 LOCATE A,B: PRINT C$
2130 NEXT I
2140 RETURN
3000 REM ****************************************
3001 REM ** SUBRUTINA TEMPERATURA CORRESPONDIENTE **
3002 REM ****************************************
3005 LET YY=Y-3
3010 LET X=32
3015 IF TC>110 THEN LET TC=110
3020 T=INT(TC+62)
3030 FOR I=1 TO 3
3040 PSET (X,YY)
3050 DRAW "M+0,-"+STR$(T)
3060 X=X+150
3070 NEXT I
3080 RETURN
6000 DATA +0,+12,+3,+3,+0,+70,+1,+3,+4,+0,+1,-3,+0,-70,+3,-3,+0,-12,-12,+0
6010 DATA +0,+12,+3,+3,+0,+70,+1,+3,+4,+0,+1,-3,+0,-70,+3,-3,+0,-12,-12,+0
6020 DATA +0,+12,+3,+3,+0,+70,+1,+3,+4,+0,+1,-3,+0,-70,+3,-3,+0,-12,-12,+0
6030 DATA 16,2,"0",16,19,"237",16,38,"32"
6040 DATA 19,1,"-18",19,19,"255",19,39,"0"
6050 DATA 4,1,"100",4,19,"373",4,37,"212"
```

Para la conversión de escalas utilizaremos la siguiente igualdad:

$$\frac{C}{100} = \frac{K-273}{100} = \frac{F-32}{180}$$

Con ello, si suponemos según la figura que tenemos 50° C vemos que habrá:

$$\frac{50}{100} = \frac{K-273}{100}$$

$$K = 323° K$$

Y la temperatura en Farenheit será:

$$\frac{50}{100} = \frac{F-32}{180}$$

$$F = 122° F$$

Adaptación a otros ordenadores

AMSTRAD

```
20 MODE 2
1005 Y = 1
1010 X = 40
1030 PLOT X,Y
1060 A = A*3:B = B*3
1070 no se pone
1080 no se pone
1090 DRAWR A,B
2005 A = 47
2020 B = Y + 54
2040 PLOT A,B
2050 DRAWR -3,0
2060 B = B + 15
3005 YY = Y + 3
3010 X = 58
3020 T = INT(TC/10*15 + 81)
3040 PLOT X,YY
3050 DRAWR 0,T
6030 DATA 1,20,"0",10,20,"237",21,20,"32"
6040 DATA 1,22,"-18",10,22,"255",22,22,"0"
6050 DATA 1,11,"100",10,11,"373",20,11,"212"
```

COMMODORE

Quitar las líneas:
350
360
370
450
460
470
550
560
570
1.000-1.130
2.000-2.140
3.000-3.080

ZX-SPECTRUM

```
20 no se pone
110 GOTO 9999
1005 LET Y = 1
1030 PLOT X,Y
1070 no se pone
1080 no se pone
1090 DRAW A,B
2020 LET B = Y + 36
2040 PLOT A,B
2050 DRAW -3,0
2060 LET B = B + 10
3005 LET YY = Y + 3
3040 PLOT X,YY
3050 DRAW 0,T
```

PROBLEMAS MATEMATICOS

Matrices

Las matrices son unas determinadas estructuras matemáticas representadas como conjuntos de números reales escritos en filas y columnas.

Dado que son operadores, se podrán sumar, restar, multiplicar, etc.

También se definirá como determinante de una matriz al valor numérico de dicha matriz. En un próximo libro aparecerá un programa para calcular determinantes.

Como aplicaciones de las matrices en el cálculo tenemos la resolución de sistemas de ecuaciones, además de la representación de vectores y operaciones con éstos. Los vectores no son más que un caso particular de las matrices que sólo tienen una fila o una columna. na.

Con el programa que aquí se presenta podremos sumar, restar y multiplicar matrices.

Comienza con la presentación de un menú en el que mediante los números 1, 2 ó 3 elegiremos la opción.

A continuación, dependiendo de la opción elegida, el programa irá a la subrutina que corresponda (suma/resta/multiplicación), en la cual se pide las dimensiones de las matrices operandos.

Habrá que tener en cuenta que, tanto en la suma como en la resta, las matrices deben tener la misma dimensión para poderse efectuar la operación.

Si esto no es así, el programa nos lo indicará con un mensaje de error y volverá al menú principal.

Si las dimensiones son correctas, el programa irá pidiendo sucesivamente los datos de cada matriz por filas.

Todas estas filas van siendo almacenadas en las matrices A y B del programa Basic.

La matriz A se lee en la subrutina que comienza en la línea 2000. La matriz B se leerá en la subrutina que comienza en 3000.

Una vez realizadas las lecturas de las dos matrices, en la subrutina que comienza en la línea 4000 se sumarán/restarán las dos matrices. Si es un producto la opción elegida, éste se realizará en la subrutina que comienza en la línea 4200.

Hay que hacer notar que la subrutina que realiza la suma es la misma que la que hace la resta. Esto se consigue cambiando de signo sucesivamente todos los elementos de la matriz B (líneas 670 a 710).

Por último, después de realizarse la operación, con sentencias INPUT se pregunta si se quiere salir del programa o no. Éste finalizará si la respuesta es afirmativa. Si no es así, el programa volverá a presentar el menú del principio.

A continuación vemos un ejemplo de una suma de matrices:

La matriz A es la siguiente:

```
FILA 1 : 1  2  3  4  5
FILA 2 : 1  2  3  4  5
FILA 3 : 1  2  3  4  5
FILA 4 : 1  2  3  4  5
FILA 5 : 1  2  3  4  5
```

Fig. 2.

La matriz B será:

```
FILA 1 : 5  4  3  2  1
FILA 2 : 5  4  3  2  1
FILA 3 : 5  4  3  2  1
FILA 4 : 5  4  3  2  1
FILA 5 : 5  4  3  2  1
```

Fig. 3.

Y la matriz suma tendrá las siguientes filas y columnas:

```
        MATRIZ SUMA

FILA 1 : 6  6  6  6  6
FILA 2 : 6  6  6  6  6
FILA 3 : 6  6  6  6  6
FILA 4 : 6  6  6  6  6
FILA 5 : 6  6  6  6  6
```

Fig. 4.

```
10 REM ***PRESENTACION***
20 SCREEN 1
30 CLS
40 PRINT TAB(8):"M A T R I C E S"
50 PRINT :PRINT:PRINT
```

```
60 PRINT TAB(7):"1 SUMA":PRINT
70 PRINT TAB(7):"2 RESTA":PRINT
80 PRINT TAB(7):"3 PRODUCTO":PRINT
90 INPUT "OPCION ELEGIDA:";A
100 IF A=1 THEN GOTO 200
110 IF A=2 THEN GOTO 500
120 IF A=3 THEN GOTO 800
130 GOTO 10
200 REM ***SUMA DE MATRICES***
210 INPUT"No. FILAS DEL SUMANDO 1:";F1
220 INPUT"No. COLUMNAS DEL SUMANDO 1:";C1
230 INPUT"No. FILAS DEL SUMANDO 2:";F2
240 INPUT"No. COLUMNAS DEL SUMANDO 2:";C2
250 IF F1=F2 AND C1=C2 THEN GOTO 320
260 CLS:PRINT :PRINT :PRINT :PRINT
270 PRINT "NO SE PUEDEN SUMAR MATRICES"
280 PRINT "DE DISTINTO NUMERO DE"
290 PRINT "FILAS O COLUMNAS"
300 FOR X=0 TO 2000:NEXT X
310 GOTO 10
320 CLS:  DIM A(F1,C1):DIM B(F2,C2)
330 PRINT "INTRODUZCA ELEMENTOS DEL SUMANDO 1":PRINT :PRINT
340 GOSUB 2000
350 PRINT "INTRODUZCA ELEMENTOS DEL SUMANDO 2":PRINT :PRINT
360 GOSUB 3000
370 CLS:PRINT TAB(8):"MATRIZ SUMA":PRINT :PRINT :PRINT
380 GOSUB 4000
390 INPUT "Pulse s para salir del programa o cualquier tecla para continuar";Z$
400 IF Z$="s" OR Z$="S" THEN END
410 CLEAR
420 GOTO 10
500 REM *** RESTA DE MATRICES***
510 INPUT"No. FILAS DEL MINUENDO :";F1
520 INPUT"No. COLUMNAS DEL MINUENDO :";C1
530 INPUT"No. FILAS DEL SUSTRAENDO :";F2
540 INPUT"No. COLUMNAS DEL SUSTRAENDO :";C2
550 IF F1=F2 AND C1=C2 THEN GOTO 620
560 CLS:PRINT :PRINT :PRINT :PRINT
570 PRINT "NO SE PUEDEN RESTAR MATRICES"
580 PRINT "DE DISTINTO NUMERO DE"
590 PRINT "FILAS O COLUMNAS"
600 FOR X=0 TO 3000:NEXT X
610 GOTO 10
620 CLS:DIM A(F1,C1):DIM B(F2,C2)
630 PRINT "INTRODUZCA ELEMENTOS DEL MINUENDO ":PRINT :PRINT
640 GOSUB 2000
650 PRINT "INTRODUZCA ELEMENTOS DEL SUSTRAENDO ":PRINT :PRINT
660 GOSUB 3000
670 FOR N=1 TO F2
680 FOR K=1 TO C2
690 LET B(N,K)=-B(N,K)
700 NEXT K
710 NEXT N
720 CLS:PRINT TAB(8):"MATRIZ DIFERENCIA":PRINT :PRINT :PRINT
730 GOSUB 4000
740 INPUT "Pulse s para salir del programa o cualquier tecla para continuar";Z$
750 IF Z$="s" OR Z$="S" THEN END
760 CLEAR
770 GOTO 10
800 REM ***MULTIPLICACION DE MATRICES***
810 INPUT"No. FILAS DEL MULTIPLICANDO :";F1
820 INPUT"No. COLUMNAS DEL MULTIPLICANDO :";C1
830 INPUT"No. FILAS DEL MULTIPLICADOR :";F2
840 INPUT"No. COLUMNAS DEL MULTIPLICADOR :";C2
850 IF C1=F2 THEN GOTO 940
860 CLS:PRINT :PRINT :PRINT :PRINT
870 PRINT "ESTAS MATRICES NO SE PUEDEN"
880 PRINT "MULTIPLICAR PORQUE EL NUMERO"
890 PRINT "DE COLUMNAS DEL MULTIPLICANDO"
900 PRINT "ES DISTINTO DEL NUMERO DE"
910 PRINT "FILAS DEL MULTIPLICADOR"
920 FOR X=0 TO 2500:NEXT X
930 GOTO 10
940 CLS:DIM A(F1,C1):DIM B(F2,C2)
950 PRINT "INTRODUZCA ELEMENTOS DEL MULTIPLICANDO ":PRINT :PRINT
960 GOSUB 2000
970 PRINT "INTRODUZCA ELEMENTOS DEL MULTIPLICADOR ":PRINT :PRINT
980 GOSUB 3000
990 CLS:PRINT TAB(8):"MATRIZ PRODUCTO":PRINT :PRINT :PRINT
1000 GOSUB 4200
1010 INPUT "Pulse s para salir del programa o cualquier tecla para continuar";Z$
1020 IF Z$="s" OR Z$="S" THEN END
1030 CLEAR
```

```
1040 GOTO 10
2000 REM ***LECTURA PRIMERA MATRIZ***
2010 FOR N=1 TO F1
2020 PRINT "FILA ";N:PRINT :PRINT
2030 FOR K=1 TO C1
2040 INPUT A(N,K)
2050 NEXT K
2060 PRINT :PRINT
2070 NEXT N
2080 CLS
2090 FOR N=1 TO F1
2100 PRINT "FILA ";N;":";
2110 FOR K=1 TO C1
2120 PRINT A(N,K);" ";
2130 NEXT K
2140 PRINT :PRINT
2150 NEXT N
2160 FOR X=1 TO 3000:NEXT X
2170 CLS
2180 RETURN
3000 REM ***LECTURA SEGUNDA MATRIZ***
3010 FOR N=1 TO F2
3020 PRINT "FILA ";N:PRINT :PRINT
3030 FOR K=1 TO C2
3040 INPUT B(N,K)
3050 NEXT K
3060 NEXT N
3070 CLS
3080 FOR N=1 TO F2
3090 PRINT "FILA ";N;":";
3100 FOR K=1 TO C2
3110 PRINT B(N,K);" ";
3120 NEXT K
3130 PRINT :PRINT
3140 NEXT N
3150 FOR X=1 TO 3000:NEXT X
3160 RETURN
4000 REM ***OPERACION SUMA/RESTA DE MATRICES***
4010 FOR N=1 TO F1
4020 PRINT"FILA ";N;":";
4030 FOR K=1 TO C1
4040 PRINT A(N,K)+B(N,K);" ";
4050 NEXT K
4060 PRINT :PRINT
4070 NEXT N
4080 RETURN
4200 REM ***OPERACION PRODUCTO DE MATRICES***
4210 FOR N=1 TO F1
4220 PRINT"FILA ";N;":";
4230 FOR F=1 TO C2
4240 LET W=0
4250 FOR K=1 TO C1
4260 LET W=(A(N,K)*B(K,F))+W
4270 NEXT K
4280 PRINT W;" ";
4290 NEXT F
4300 PRINT :PRINT
4310 NEXT N
4320 RETURN
```

Variaciones para otros ordenadores

AMSTRAD

20 MODE 1
410 no se pone
760 no se pone
1030 no se pone

MSX

20 SCREEN 0
410 no se pone
760 no se pone
1030 no se pone

ZX-SPECTRUM

20 no se pone
400 ...THEN GOTO 9999
410 no se pone
750 ...THEN GOTO 9999
760 no se pone
1020 ...THEN GOTO 9999
1030 no se pone

COMMODORE

20 no se pone
260 PRINT CHR$(147) por CLS
320 PRINT CHR$(147) por CLS
370 PRINT CHR$(147) por CLS
410 no se pone
560 PRINT CHR$(147) por CLS
620 PRINT CHR$(147) por CLS
720 PRINT CHR$(147) por CLS

```
760 no se pone
860 PRINT CHR$(147) por CLS
940 PRINT CHR$(147) por CLS
990 PRINT CHR$(147) por CLS
1030 no se pone
2080 PRINT CHR$(147) por CLS
2170 PRINT CHR$(147) por CLS
3070 PRINT CHR$(147) por CLS
```

SOCIEDAD

La conjugación verbal

En el programa que a continuación presentamos están definidas las conjugaciones de los verbos regulares de la lengua española.

Como todos sabemos, en nuestro idioma, a diferencia de otras lenguas también de origen romance, la terminación verbal es lo que caracteriza todos los tiempos y todas las personas de los verbos; en el castellano se puede prescindir de los pronombres personales que acompañan a las personas de los verbos; solamente con los morfemas que se les unen a las raíces podemos determinar cuál es el modo, el tiempo y la persona de la forma que se analice.

Hemos enunciado todas las formas impersonales y las formas personales en sus tres modos: indicativo, subjuntivo e imperativo.

Con este programa se podrá obtener cualquier tiempo verbal a partir de la introducción del verbo en infinitivo.

Unicamente habrá que tener en cuenta que el verbo que introduzcamos debe ser regular, ya que si no lo es, el programa no funcionaría correctamente.

Indagando en la construcción del presente programa, se puede ver en la línea 180 el dimensionamiento de tres matrices de tres dimensiones cada una, a saber:

I$ para almacenar todos los tiempos de indicativo

S$ para almacenar el modo subjuntivo

O$ para almacenar el presente de imperativo.

La primera dimensión determina la conjugación, la segunda el tiempo verbal y la tercera la persona. Así, en las líneas 190 a 370 se van leyendo sucesivamente todas las desinencias verbales y se van almacenando en estas matrices para después concatenarlas con la raíz del verbo introducido por el usuario para presentar en pantalla el tiempo que se ha elegido.

El programa consta de un menú donde elegiremos con un número del 1 al 4 el modo verbal, después de haber introducido el verbo a conjugar, en infinitivo y en mayúsculas.

A continuación, dependiendo de la opción elegida, aparecerá en pantalla otro menú (a excepción de las formas no personales que aparecen directamente), donde eligiremos el tiempo verbal que queremos que sea presentado.

Al final de todo esto el ordenador nos hará dos preguntas acerca de si queremos obtener otro tiempo verbal y a continuación si queremos conjugar otro verbo. Todo esto se hace en la subrutina FINAL DEL PROGRAMA (línea 1890).

La clave de todo el programa la tenemos en las líneas 380 a 440 en las cuales almacenamos en sendas variables alfanuméricas la raíz y la desinencia del verbo introducido. A partir de esta última obtendremos la conjugación de que se trata.

```
1 SCREEN 1
10 REM ***VERBOS REGULARES***
20  REM ***PRESENTACION***
30 CLS
40 PRINT :PRINT :PRINT :PRINT :PRINT
50 PRINT TAB(10);"CONJUGACION":PRINT :PRINT
60 PRINT TAB(13);"DE":PRINT :PRINT
70 PRINT TAB(8);"VERBOS REGULARES"
80 FOR F=1 TO 2000 :NEXT F
90 REM ***LECTURA PRESENTACION TIEMPOS VERBALES***
100 DIM T$(10)
110 FOR A=1 TO 10
120 READ T$(A)
130 NEXT A
140 DIM U$(6)
150 FOR A=1 TO 6
160 READ U$(A)
170 NEXT A
180 DIM I$(3,10,6):DIM S$(3,6,6):DIM O$(3,1,3)
190 FOR E=1 TO 3
200 FOR A=1 TO 10
210 FOR C=1 TO 6
```

```
220 READ I$(E,A,C)
230 NEXT C
240 NEXT A
250 NEXT E
260 FOR E=1 TO 3
270 FOR A=1 TO 6
280 FOR C=1 TO 6
290 READ S$(E,A,C)
300 NEXT C
310 NEXT A
320 NEXT E
330 FOR E=1 TO 3
340 FOR C=1 TO 3
350 READ O$(E,1,C)
360 NEXT C
370 NEXT E
380 CLS:INPUT "VERBO A CONJUGAR (EN INFINITIVO):";X$
390 LET L=LEN (X$)
400 LET H$=LEFT$(X$,L-2)
410 LET Z$=RIGHT$(X$,2)
420 IF Z$="AR" THEN LET W=1:LET P$="ADO"
430 IF Z$="ER" THEN LET W=2:LET P$="IDO"
440 IF Z$="IR" THEN LET W=3:LET P$="IDO"
450 REM ***MENU DE MODOS VERBALES***
460 CLS: PRINT TAB(10):"M O D O S"
470 PRINT :PRINT :PRINT :PRINT
480 PRINT TAB(4):"1 FORMAS NO PERSONALES":PRINT :PRINT
490 PRINT TAB(8);"2 INDICATIVO":PRINT :PRINT
500 PRINT TAB(8);"3 SUBJUNTIVO":PRINT :PRINT
510 PRINT TAB(8);"4 IMPERATIVO":PRINT :PRINT
520 INPUT "OPCION ELEGIDA:";B
530 IF B<1 OR B>4 THEN GOTO 450
540 CLS
550 IF B=1 THEN GOTO 590
560 IF B=2 THEN GOTO 720
570 IF B=3 THEN GOTO 990
580 IF B=4 THEN GOTO 1220
590 REM ***FORMAS NO PERSONALES***
600 PRINT TAB(10);"F O R M A S":PRINT :PRINT
610 PRINT TAB(4);"N O  P E R S O N A L E S":PRINT :PRINT
620 PRINT TAB(10);"SIMPLES":PRINT
630 PRINT "INFINITIVO......";X$:PRINT
640 IF W=1 THEN LET Q$="ANDO":GOTO 660
650 LET Q$="IENDO"
660 PRINT "GERUNDIO......";H$;Q$:PRINT
670 PRINT "PARTICIPIO......";H$;P$:PRINT:PRINT
680 PRINT TAB(8);"COMPUESTAS":PRINT
690 PRINT "INFINITIVO......";"HABER ";H$;P$:PRINT
700 PRINT "GERUNDIO......";"HABIENDO ";H$;P$:PRINT
710 GOTO 1900
720 REM ***INDICATIVO***
730 CLS: PRINT TAB(6):"I N D I C A T I V O"
740 PRINT :PRINT :PRINT
750 PRINT TAB(5):"1 PRESENTE"
760 PRINT TAB(5);"2 PRET. IMPERFECTO"
770 PRINT TAB(5):"3 PRET. INDEFINIDO"
780 PRINT TAB(5);"4 FUTURO IMPERFECTO"
790 PRINT TAB(5):"5 CONDICIONAL SIMPLE"
800 PRINT TAB(5);"6 PRET. PERFECTO"
810 PRINT TAB(5):"7 PRET. PLUSCUAMPERFECTO"
820 PRINT TAB(5);"8 PRET. ANTERIOR"
830 PRINT TAB(5);"9 FUTURO PERFECTO"
840 PRINT TAB(4);"10 CONDICIONAL COMPUESTO"
850 INPUT "OPCION ELEGIDA:";N
860 IF N<1 OR N>10 THEN GOTO 730
870 CLS
880 PRINT TAB(5):T$(N):PRINT :PRINT :PRINT :PRINT
890 GOSUB 1820
900 IF N>5 THEN GOTO 950
910 FOR X=1 TO 6
920 LOCATE 5+X,14:PRINT H$;I$(W,N,X)
930 NEXT X
940 GOTO 1900
950 FOR X=1 TO 6
960 LOCATE 5+X,11:PRINT I$(W,N,X);" ";H$;P$
970 NEXT X
980 GOTO 1900
990 REM ***SUBJUNTIVO***
1000 CLS: PRINT TAB(6):"S U B J U N T I V O"
1010 PRINT :PRINT :PRINT
1020 PRINT TAB(5);"1 PRESENTE"
1030 PRINT TAB(5);"2 PRET. IMPERFECTO"
1040 PRINT TAB(5):"3 FUTURO IMPERFECTO"
```

```
1050 PRINT TAB(5);"4 PRET. PERFECTO"
1060 PRINT TAB(5);"5 PRET. PLUSCUAMPERFECTO"
1070 PRINT TAB(5);"6 FUTURO PERFECTO"
1080 INPUT "OPCION ELEGIDA:";N
1090 IF N<1 OR N>6 THEN GOTO 1000
1100 CLS
1110 PRINT TAB(5):U$(N):PRINT :PRINT :PRINT :PRINT
1120 GOSUB 1820
1130 IF N>3 THEN GOTO 1180
1140 FOR X=1 TO 6
1150 LOCATE 5+X,14:PRINT H$;S$(W,N,X)
1160 NEXT X
1170 GOTO 1900
1180 FOR X=1 TO 6
1190 LOCATE 5+X,11:PRINT S$(W,N,X);" ";H$;P$
1200 NEXT X
1210 GOTO 1900
1220 REM ***IMPERATIVO***
1230 PRINT TAB(5);"PRESENTE":PRINT :PRINT :PRINT :PRINT
1240 FOR X=1 TO 3
1250 LOCATE 5+X,14:PRINT H$;O$(W,1,X)
1260 NEXT X
1270 GOTO 1900
1280 DATA PRESENTE,PRETERITO IMPERFECTO,PRETERITO INDEFINIDO,FUTURO IMPERFECTO,C
ONDICIONAL SIMPLE,PRETERITO PERFECTO,PRETERITO PLUSCUAMPERFECTO,PRETERITO ANTERI
OR,FUTURO PERFECTO,CONDICIONAL COMPUESTO
1290 DATA PRESENTE,PRETERITO IMPERFECTO,FUTURO IMPERFECTO,PRETERITO PERFECTO,PRE
TERITO PLUSCUAMPERFECTO,FUTURO PERFECTO
1300 REM ***LECTURA TERMINACIONES VERBALES***
1310 DATA O,AS,A,AMOS,AIS,AN
1320 DATA ABA,ABAS,ABA,ABAMOS,ABAIS,ABAN
1330 DATA E,ASTE,O,AMOS,ASTEIS,ARON
1340 DATA ARE,ARAS,ARA,AREMOS,AREIS,ARAN
1350 DATA ARIA,ARIAS,ARIA,ARIAMOS,ARIAIS,ARIAN
1360 DATA HE,HAS,HA,HEMOS,HABEIS,HAN
1370 DATA HABIA,HABIAS,HABIA,HABIAMOS,HABIAIS,HABIAN
1380 DATA HUBE,HUBISTE,HUBO,HUBIMOS,HUBISTEIS,HUBIERON
1390 DATA HABRE,HABRAS,HABRA,HABREMOS,HABREIS,HABRAN
1400 DATA HABRIA,HABRIAS,HABRIA,HABRIAMOS,HABRIAIS,HABRIAN
1410 DATA O,ES,E,EMOS,EIS,EN
1420 DATA IA,IAS,IA,IAMOS,IAIS,IAN
1430 DATA I,ISTE,IO,IMOS,ISTEIS,IERON
1440 DATA ERE,ERAS,ERA,EREMOS,EREIS,ERAN
1450 DATA ERIA,ERIAS,ERIA,ERIAMOS,ERIAIS,ERIAN
1460 DATA HE,HAS,HA,HEMOS,HABEIS,HAN
1470 DATA HABIA,HABIAS,HABIA,HABIAMOS,HABIAIS,HABIAN
1480 DATA HUBE,HUBISTE,HUBO,HUBIMOS,HUBISTEIS,HUBIERON
1490 DATA HABRE,HABRAS,HABRA,HABREMOS,HABREIS,HABRAN
1500 DATA HABRIA,HABRIAS,HABRIA,HABRIAMOS,HABRIAIS,HABRIAN
1510 DATA O,ES,E,IMOS,IS,EN
1520 DATA IA,IAS,IA,IAMOS,IAIS,IAN
1530 DATA I,ISTE,IO,IMOS,ISTEIS,IERON
1540 DATA IRE,IRAS,IRA,IREMOS,IREIS,IRAN
1550 DATA IRIA,IRIAS,IRIA,IRIAMOS,IRIAIS,IRIAN
1560 DATA HE,HAS,HA,HEMOS,HABEIS,HAN
1570 DATA HABIA,HABIAS,HABIA,HABIAMOS,HABIAIS,HABIAN
1580 DATA HUBE,HUBISTE,HUBO,HUBIMOS,HUBISTEIS,HUBIERON
1590 DATA HABRE,HABRAS,HABRA,HABREMOS,HABREIS,HABRAN
1600 DATA HABRIA,HABRIAS,HABRIA,HABRIAMOS,HABRIAIS,HABRIAN
1610 DATA E,ES,E,EMOS,EIS,EN
1620 DATA ARA,ARAS,ARA,ARAMOS,ARAIS,ARAN
1630 DATA ARE,ARES,ARE,AREMOS,AREIS,AREN
1640 DATA HAYA,HAYAS,HAYA,HAYAMOS,HAYAIS,HAYAN
1650 DATA HUBIERA,HUBIERAS,HUBIERA,HUBIERAMOS,HUBIERAIS,HUBIERAN
1660 DATA HUBIERE,HUBIERES,HUBIERE,HUBIEREMOS,HUBIEREIS,HUBIEREN
1670 DATA A,AS,A,AMOS,AIS,AN
1680 DATA IERA,IERAS,IERA,IERAMOS,IERAIS,IERAN
1690 DATA IERE,IERES,IERE,IEREMOS,IEREIS,IEREN
1700 DATA HAYA,HAYAS,HAYA,HAYAMOS,HAYAIS,HAYAN
1710 DATA HUBIERA,HUBIERAS,HUBIERA,HUBIERAMOS,HUBIERAIS,HUBIERAN
1720 DATA HUBIERE,HUBIERES,HUBIERE,HUBIEREMOS,HUBIEREIS,HUBIEREN
1730 DATA A,AS,A,AMOS,AIS,AN
1740 DATA IERA,IERAS,IERA,IERAMOS,IERAIS,IERAN
1750 DATA IERE,IERES,IERE,IEREMOS,IEREIS,IEREN
1760 DATA HAYA,HAYAS,HAYA,HAYAMOS,HAYAIS,HAYAN
1770 DATA HUBIERA,HUBIERAS,HUBIERA,HUBIERAMOS,HUBIERAIS,HUBIERAN
1780 DATA HUBIERE,HUBIERES,HUBIERE,HUBIEREMOS,HUBIEREIS,HUBIEREN
1790 DATA A TU,EMOS NOSOTROS,AD VOSOTROS
1800 DATA E TU,AMOS NOSOTROS,ED VOSOTROS
1810 DATA E TU,AMOS NOSOTROS,ID VOSOTROS
1820 LOCATE 6,3:PRINT "YO"
1830 LOCATE 7,3:PRINT "TU"
1840 LOCATE 8,3:PRINT "EL"
```

```
1850 LOCATE 9,1:PRINT "NOSOTROS"
1860 LOCATE 10,1:PRINT "VOSOTROS"
1870 LOCATE 11,2:PRINT "ELLOS"
1880 RETURN
1890 REM ***FINAL DEL PROGRAMA***
1900 INPUT "OTRO TIEMPO (S/N)";E$
1910 IF E$="S" OR E$="s" THEN GOTO 460
1920 INPUT "OTRO VERBO (S/N)";V$
1930 IF V$="S" OR V$="s" THEN GOTO 380
1940 END
```

Variaciones para otros ordenadores

AMSTRAD

1 MODE 1
CAMBIAR LAS SENTENCIAS LOCATE X,Y POR LOCATE Y,X

COMMODORE

1 no se pone
30 PRINT CHR$(147) 730 PRINT CHR$(147)
380 PRINT CHR$(147)870 PRINT CHR$(147)
460 PRINT CHR$(147)1000 PRINT CHR$(147)
540 PRINT CHR$(147)1100 PRINT CHR$(147)

Para las sentencias LOCATE ver simulación de ésta en tomo 1

ZX-SPECTRUM

1 no se pone 400 LET H$=X$(TO LEN X$-2)
390 no se pone 410 LET Z$=X$(LEN X$-1 TO)
1940 GOTO 9999

Todos los elementos de las sentencias DATA deberán ir entre comillas.

Todas las sentencias LOCATE X, Y: PRINT ... se sustituyen por PRINT AT X, Y;...

Las matrices alfanuméricas necesitan otra dimensión.

MSX

Valen las mismas modificaciones que para el Amstrad, a excepción de la línea 1, que se sustituirá por:

1 SCREEN 0

PARA LOS MAS PEQUEÑOS

Conjuntos

En esta ocasión hemos tenido que contener las ganas de entrecomillar de forma irónica el título de esta sección. Sí, creemos que "Para los más jóvenes", en este caso, no va dedicado sólo a ellos.

Es ya bien conocida la gran revolución que supusieron los conjuntos en la matemática moderna y cómo algunas personas, ya no tan jóvenes, se sentían incapaces de adaptar su mentalidad y trabajar con los "conjuntitos".

No desperdiciéis esta ocasión, ni tú, "más joven", que estudias en este momento los conjuntos, ni tú, "menos joven", que nunca los has estudiado y siempre has querido descifrarlos.

Con este programa podrás aprender las bases de los conjuntos, cómo hallar su unión, su intersección, y cómo calcular el cardinal de un conjunto.

A grandes rasgos, aunque suponemos la teoría conocida, vamos a introducirte estos conceptos:

— Cardinal de un conjunto es su número de elementos.

— Intersección de dos conjuntos es el conjunto formado por los elementos comunes a los dos conjuntos.

— Unión de dos conjuntos es el conjunto formado por todos los elementos que hay en los dos conjuntos.

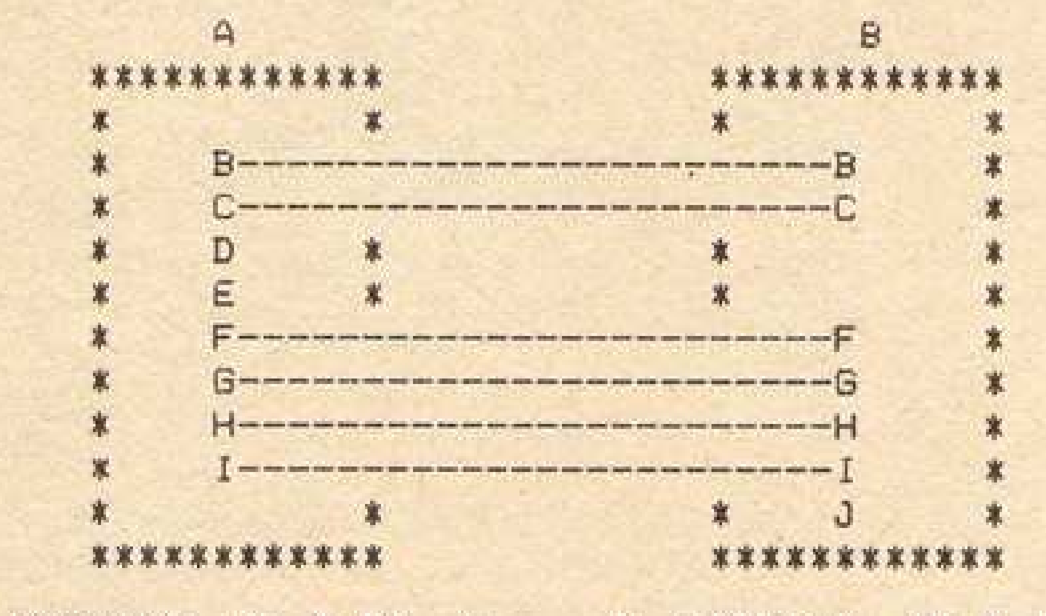


Fig. 5.



Fig. 6.



Fig. 7.

```
10 REM ************************************
20 REM **     PROGRAMA DE CONJUNTOS      **
30 REM ************************************
40 LET A1=10:LET B1=10:LET C1=10
50 REM ********************************************
60 REM ** DIMENSIONANDO LAS MATRICES UTILIZADAS **
70 REM ********************************************
80 DIM A$(A1,B1): DIM C(C1):DIM B$(C1):DIM R$(C1)
90 CLS
100 RANDOMIZE TIMER
110 FOR J=1 TO 10
120 REM ***********************************
130 REM ** LEYENDO LAS MATRICES DE DATOS **
140 REM ***********************************
150 FOR I=1 TO 10
160 READ A$(I,J)
170 NEXT I
180 NEXT J
190 FOR I=1 TO 10
200 READ C(I)
210 NEXT I
220 REM **************************************
230 REM ** GENERANDO CONJUNTOS ALEATORIOS **
240 REM **************************************
250 LET N1=INT (RND*10)+1
260 LET N2=INT (RND*10)+1
270 GOSUB 1000
280 REM *****************************
290 REM ** PREGUNTAS DE CARDINALES **
300 REM *****************************
310 INPUT "CARDINAL DE A";A
320 INPUT "CARDINAL DE B";B
330 IF A<>C(N1) THEN PRINT "EL CARDINAL DE A ES ";C(N1)
340 IF B<>C(N2) THEN PRINT "EL CARDINAL DE B ES ";C(N2)
350 REM *******************************
360 REM ** PREGUNTAS DE INTERSECCION **
370 REM *******************************
380 LET C=0
390 GOSUB 2000
400 PRINT "¿CUAL ES LA INTERSECCION"
410 FOR I=1 TO D
420 INPUT R$(I)
430 IF R$(I)<>B$(I) THEN LET C=1
440 NEXT I
450 CLS
460 GOSUB 3000
470 INPUT "CARDINAL DE LA INTERSECCION";Q
480 IF Q<>D THEN PRINT "EL CARDINAL DE LA INTERSECCION ES ";D
490 CLS
500 GOSUB 1000
510 REM ***************************
520 REM ** PREGUNTAS DE LA UNION **
530 REM ***************************
540 LET C=0
550 GOSUB 4000
560 PRINT "CUAL ES LA UNION"
570 FOR I=1 TO D
580 INPUT R$(I)
590 IF R$(I)<>B$(I) THEN LET C=1
600 NEXT I
610 CLS
620 GOSUB 5000
630 INPUT "CARDINAL DE LA UNION";Q
640 IF Q<>D THEN PRINT "EL CARDINAL DE LA UNION ES ";D
650 INPUT "¿QUIERES INTENTARLO OTRA VEZ";R$
660 IF R$="S" OR R$="s" THEN RUN
670 END
1000 REM ***************************************
1010 REM ** SUBRUTINA DE DIBUJO DE CONJUNTOS **
1020 REM ***************************************
1030 PRINT TAB(10);"     A      ";TAB(35);"      B     "
1040 PRINT TAB(10);"*************";TAB(35);"*************"
1050 FOR I=1 TO 10
1060 IF A$(I,N1)=" " OR A$(I,N2)=" " THEN LET S=0
1070 PRINT TAB(10);"*";TAB(15);A$(I,N1);
1080 IF S=1 THEN PRINT "--------------------------";
1090 IF S=0 THEN PRINT TAB(21)"*";TAB(35);"*";
1100 PRINT TAB(40);A$(I,N2);TAB(46);"*"
1110 LET S=1
1120 NEXT I
1130 PRINT TAB(10);"*************";TAB(35);"*************"
1140 PRINT
```

```
1150 RETURN
2000 REM ****************************
2010 REM ** SUBRUTINA INTERSECCION **
2020 REM ****************************
2030 LET D=0
2040 FOR I=1 TO 10
2050 IF A$(I,N1)=" " THEN GOTO 2090
2060 IF A$(I,N2)=" " THEN GOTO 2090
2070 LET B$(D+1)=A$(I,N1)
2080 LET D=D+1
2090 NEXT I
2100 RETURN
3000 REM ***********************************
3010 REM ** SUBRUTINA DIBUJO INTERSECCION **
3020 REM ***********************************
3030 PRINT TAB(17);"    INTERSECCION    "
3040 PRINT TAB(17);"********************"
3050 FOR I=1 TO 10
3060 PRINT TAB(17);"*";TAB(26);B$(I);TAB(36);"*"
3070 NEXT I
3080 PRINT TAB(17);"********************"
3090 PRINT
3100 IF C=1 THEN PRINT "LO HAS HECHO MAL"
3110 RETURN
4000 REM *********************
4010 REM ** SUBRUTINA UNION **
4020 REM *********************
4030 LET D=0
4040 FOR I=1 TO 10
4050 IF A$(I,N1)<>" " THEN LET B$(D+1)=A$(I,N1):GOTO 4080
4060 IF A$(I,N2)<>" " THEN LET B$(D+1)=A$(I,N2):GOTO 4080
4070 GOTO 4090
4080 LET D=D+1
4090 NEXT I
4100 RETURN
5000 REM ****************************
5010 REM ** SUBRUTINA DIBUJO UNION **
5020 REM ****************************
5030 PRINT TAB(17);"        UNION       "
5040 PRINT TAB(17);"********************"
5050 FOR I=1 TO 10
5060 PRINT TAB(17);"*";TAB(26);B$(I);TAB(36);"*"
5070 NEXT I
5080 PRINT TAB(17);"********************"
5090 PRINT
5100 IF C=1 THEN PRINT "LO HAS HECHO MAL"
5110 RETURN
6000 DATA " ","B","C","D","E","F","G","H","I"," "
6010 DATA " "," ","C","D","E"," ","G"," "," "," "
6020 DATA " ","B","C","D"," ","F","G","H","I","J"
6030 DATA " ","B","C"," ","E","F","G","H","I"," "
6040 DATA " "," ","C","D","E"," "," "," "," "," "
6050 DATA " ","B","C"," "," ","F","G","H","I","J"
6060 DATA " ","B"," "," "," "," ","G","H","I","J"
6070 DATA "A"," ","C"," ","E"," ","G"," ","I"," "
6080 DATA " ","B"," ","D","E","F"," "," "," "," "
6090 DATA " ","B","C"," "," "," ","G"," "," "," "
6100 DATA 8,4,8,7,3,7,5,5,4,3
```

Variaciones para MSX

100 no se pone.
250 LET N1= INT(RND(1)*10)+1
260 LET N2= INT(RND(1)*10)+1

Variaciones para ZX-SPECTRUM

100 no se pone. 670 GOTO 9999

Variaciones para COMMODORE

90 PRINT CHR$(147)
100 no se pone.
250 LET N1=INT(RND(1)*10)+1
260 LET N2=INT(RND(1)*10)+1
450 PRINT CHR$(147)
490 PRINT CHR$(147)
610 PRINT CHR$(147)

Sobre el programa

Está programado en un BASIC muy general, que te será muy fácil comprender.

Consta de un programa principal que va llamando a las subrutinas siguientes:

— Subrutina de dibujo en la pantalla de los dos conjuntos iniciales (1000).

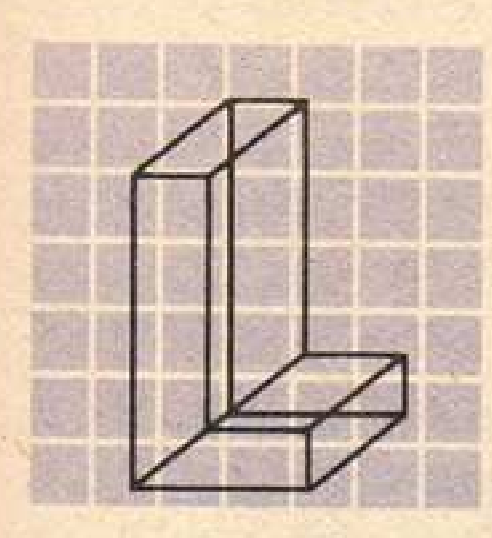
— Subrutina de cálculo del conjunto intersección (2000).

— Subrutina de dibujo en la pantalla del conjunto intersección (3000).

— Subrutina de cálculo del conjunto unión (4000).

— Subrutina de dibujo en la pantalla del conjunto unión (5000).

Los lenguajes de programación



LAS primeras máquinas electrónicas de cálculo, antecesoras de los ordenadores actuales, podían programarse. Sin embargo, la programación era muy rudimentaria y consistía en desembornar cables de sus posiciones y conectarlos en puntos diferentes (esto ocurría, por ejemplo, en el ENIAC). La tarea era, pues, francamente complicada, y sólo se llevaba a cabo para cálculos importantes (largos en tiempo).

Una mejora considerable consistió en programar las máquinas introduciendo las instrucciones en fichas perforadas. El sistema era mucho más cómodo (ya que permitía reordenar las fichas, en lugar de sacar cables y cambiarlos de posición, etc.). Sin embargo, las instrucciones se daban al ordenador en lenguaje máquina (y mediante ceros y unos binarios), por lo que la programación resultaba muy complicada y sólo posible para aquellas personas iniciadas (que debían conocer, además, cada modelo de máquina "al dedillo", ya que existían —y aún existen— diferencias considerables entre una máquina y otra).

(El lector puede intentar imaginar el trabajo desarrollado al introducir al ordenador instrucciones complejas en "lenguaje máquina" de sólo ceros y unos.)

Los lenguajes de programación o lenguajes simbólicos nacen de la necesidad de que el hombre se comunique con la máquina; su fin primordial es facilitar esa comunicación.

En los años 50 aparecen los primeros lenguajes de programación. Consistían en un número limitado de palabras y símbolos. Su aportación esencial era la concisión, es decir, una única instrucción en un lenguaje de programación supone gran número de instrucciones en "lenguaje máquina". (Más tarde, el compilador es el que hace el trabajo de traducir a lenguaje máquina las instrucciones del lenguaje de que se trate.)

Estos primeros programas, que facilitan el intercambio de información entre máquina y hombre, se llaman "ensambladores". Asociaban palabras mnemotécnicas para indicar determinados pasos en la máquina (por ejemplo, MVC para mandar el traslado de un dato en-

¿Sabía usted que...

En una de las múltiples guerras franco-españolas los españoles utilizaban para mandar sus mensajes una clave muy complicada. Esta clave utilizaba casi 600 caracteres y se cambiaba periódicamente, resultando muy complicada de utilizar, aunque prácticamente parecía imposible de descifrar. Enrique IV de Francia, hombre práctico (tuvo que convertirse al catolicismo para aceptar el trono de su país) conoció a un abogado y matemático aficionado, François Vieta (1540-1603), que tenía fama de resolver problemas complejos matemáticamente, más por afición que por dedicación. El rey le entregó su confianza, presentándole un despacho interceptado y pidiéndole que lo descifrara. Vieta consiguió descifrarlo, utilizando un algoritmo de aproximaciones sucesivas. A partir de ese momento los franceses lograron descifrar la mayoría de los despachos, sacando enorme provecho de ello. El rey de España, Felipe II, jamás pudo sospechar lo ocurrido, y (desde luego sólo lo dice la leyenda negra) se quejó al Papa, alegando que los franceses realizaban hechicerías.

tre varios puntos de la memoria), en lugar de tener que escribir la instrucción completa mediante los ceros y unos binarios (o en hexadecimal).

Sin embargo, es difícil programar en ensamblador sin conocer de forma absoluta la construcción lógica del ordenador, es decir, sin dominar la máquina hasta sus últimos detalles, y esto hace que cada especialista lo sea de una máquina o gama de máquinas en particular.

Los constructores del ordenador proporcionan el "logicial de base", es decir, el conjunto de instrucciones que hacen que el ordenador "se pueda utilizar". En los ordenadores grandes este "logicial de base" puede estar compuesto por millones de instrucciones.

En seguida se vio la necesidad de utilizar lenguajes más sofisticados para facilitar la tarea de los ordenadores. Así, se empezó a hablar de lenguajes de alto nivel frente a los lenguajes ensambladores.

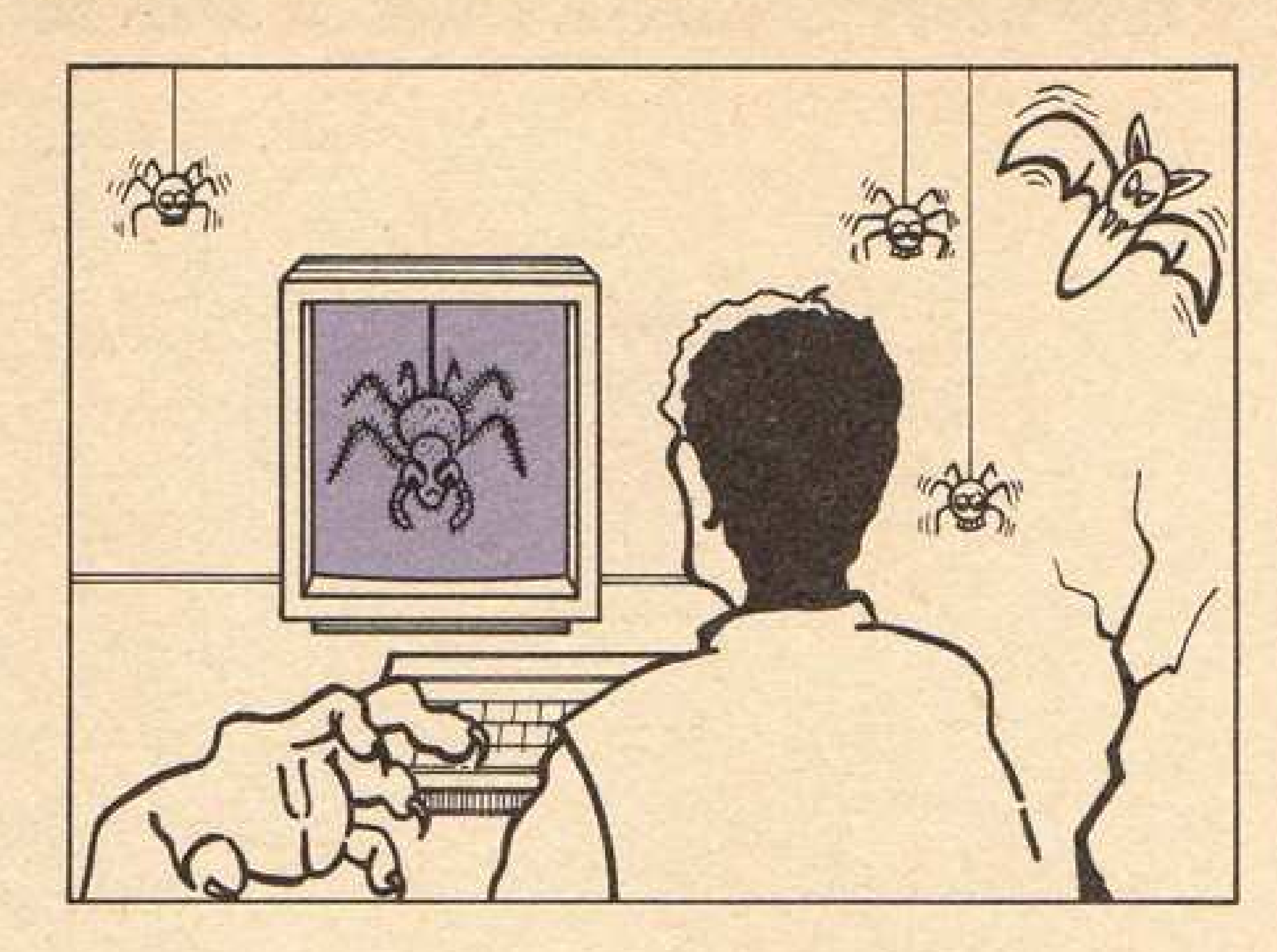
En 1954 la IBM desarrolla un nuevo lenguaje con fines científicos: el FORTRAN (fórmula translator-traductor de fórmulas). Las primeras versiones de ese lenguaje tienen ciertos problemas que se van solventando en las versiones siguientes, pero el lenguaje se va decantando cada vez más a ser utilizado por aquellos usuarios cuyos problemas sean principalmente científicos.

En 1956 el Departamento de Defensa de los Estados Unidos decide adoptar un único lenguaje de programación, que servirá, además, para todos los ordenadores de la Administración. Y nace el COBOL (Common Business Oriented Language-Lenguaje Común Orientado a los Negocios). En su desarrollo también colaboraron algunos constructores de ordenadores y usuarios importantes.

Sin embargo, los lenguajes de programación de que hemos hablado no son perfectos. Y poco a poco se van desarrollando otros lenguajes, intentando resolver los problemas que presentaban los lenguajes anteriores.

En 1960 aparece el APL (A Programming Language-Un Lenguaje de Programación), y el ALGOL (Algorythm Language-Lenguaje Algorítmico). Estos dos lenguajes se utilizan generalmente para desarrollar temas científicos.

Realmente, el ALGOL nace de una reunión en Zurich (Suiza) en la que participan la American Asociation for Computer Machines y la Asociación Europea de Mecánica y Matemáticas Aplicadas. El objetivo de la reunión es ponerse a trabajar en el desarrollo de un lenguaje común, simbólico para cálculos científicos. El lenguaje, que intenta ser universal, se llamará ALGOL. La primera versión aparece en el año 1960 (ALGOL 60).



¿Sabía usted que...

En 1962 se concedió el premio Nobel de Química al profesor Kendrew, de Cambridge, por sus trabajos sobre la molécula de mioglobina. La mioglobina es una proteína bastante compleja (no tanto como la hemoglobina), y para su estudio era necesario procesar gran cantidad de información.

Su importancia en los seres vivos es grande, ya que es capaz de unirse libremente a una molécula de oxígeno y pasar a almacenar este elemento en el tejido muscular.

Naturalmente, químicos, médicos y otros científicos se habían interesado en el ordenador desde sus comienzos, y en este tipo de investigaciones resultaba enormemente útil.

En investigación de Química Orgánica, por ejemplo, la ayuda de un ordenador es fundamental, ya que las cadenas de átomos que forman las moléculas orgánicas son muy complejas, pues ofrecen innumerables posibilidades que deben ser analizadas una a una. Además, en Química Orgánica es absolutamente necesario conocer la disposición espacial de la molécula para comprender su comportamiento (sobre todo, en estas moléculas tan enormes). Como la molécula tiene varios miles de átomos, Kendrew tardó más de dos años en ir concretando su estructura. Trabajó en el ordenador de la Univesidad, el EDSAC. Más adelante, con otros ordenadores más modernos y potentes, se han llegado a conocer muchos más detalles específicos de la molécula de mioglobina.

El objetivo básico de estos lenguajes es liberar al programador de preocuparse de las condiciones especiales de cada ordenador, y permitirle escribir programas de la forma que mejor le convenga. La anterior es una buena definición de "Lenguaje universal"; es decir, un lenguaje que no necesite referirse a cada ordenador concreto. (Recordamos al lector que el compilador correspondiente traducirá el lenguaje simbólico universal al lenguaje propio del ordenador específico de que se trate.)

Los lenguajes de orientación científica deben poder utilizar y representar las notaciones matemáticas, además de no necesitar apenas explicaciones adicionales. También deben ser capaces de describir procesos de cómputo, y traducir los procesos mecánicos a programas en lenguaje máquina.

En los primeros años de su aparición el ALGOL toma un cierto auge en Europa, introduciéndose en los países del Este, y tomando especial auge en Rusia.

Sin embargo, su estructura gramatical es muy rígida, y su vocabulario es muy escaso. Estas razones hacen que su utilización se vaya estabilizando, e incluso decrezca con la aparición de otros lenguajes más modernos. Además, comercialmente, no tiene mucha utilidad, ya que es muy difícil de utilizar por personal no muy especializado.

En los años posteriores aparecen otros lenguajes, como el RPG (Report Program Generator-Generador de Programas de Informes). Había sido desarrollado por IBM para sus máquinas de tarjetas perforadas, y su objetivo era ayudar a producir fácilmente informes. Las tarjetas y tubos quedaron obsoletas, pero el RPG se sigue manteniendo, orientado a problemas de aplicación general.

Sin embargo, todos los lenguajes de programación antes citados, y que hoy en día siguen utilizándose, son muy herméticos, y sólo son útiles para fines concretos, utilizados por personal muy técnico.

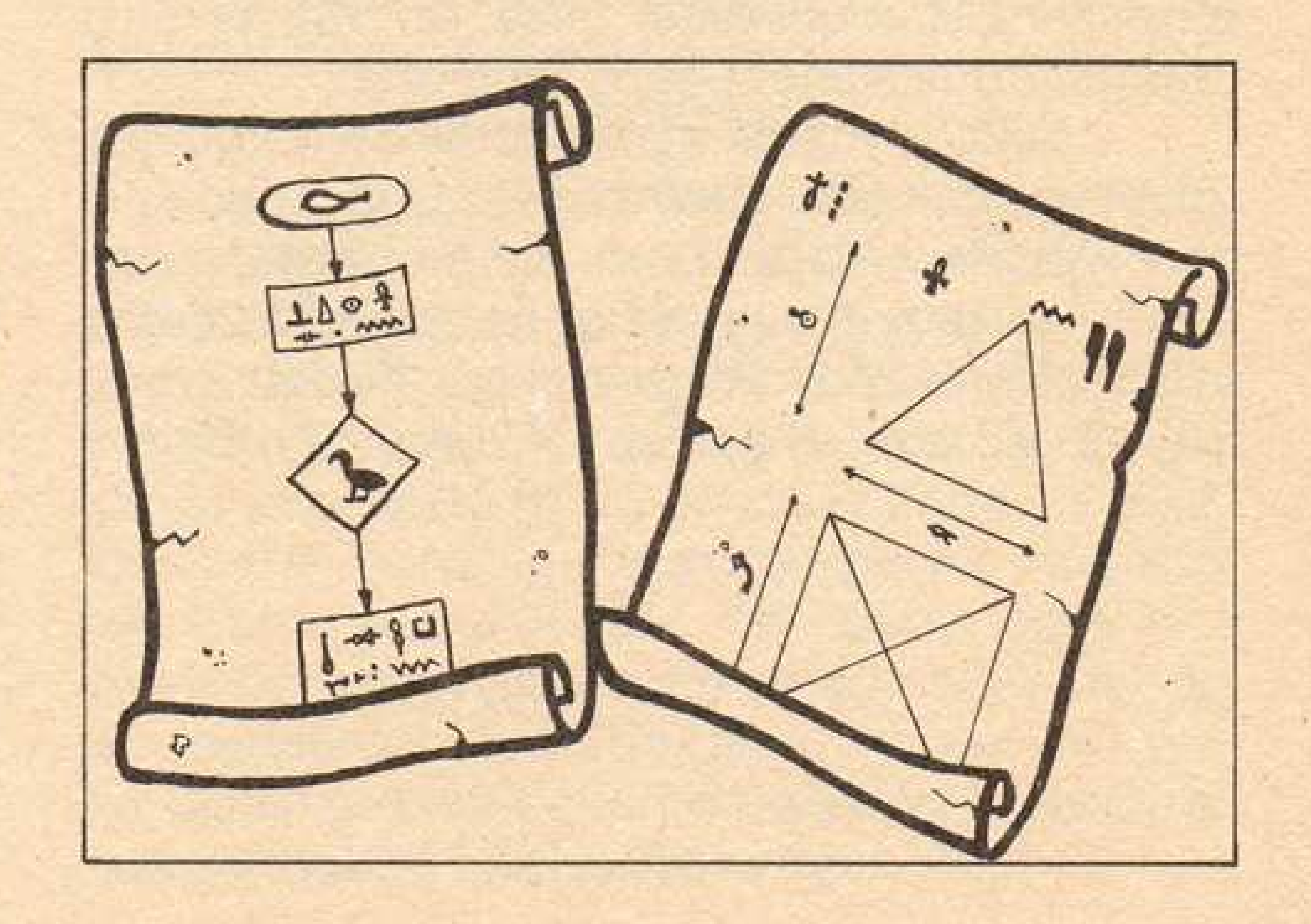
El primer lenguaje pensado para "iniciar" a los estudiantes en el mundo fascinante de la programación es el BASIC (Beginners All Purpose Symbolic Instruction Code -Código de Instrucciones Simbólicas de Propósito General para Principiantes), y para ello hubo que esperar a 1964. Fue desarrollado en la Universidad de Dartmouth (EE.UU.) durante los años 60, ya que la Universidad había tomado la decisión de enseñar informática a todos sus alumnos (700 cada nuevo curso).

El éxito del BASIC fue enorme. Era sencillo de aprender, y las instrucciones se basaban en palabras comunes inglesas. Los que lo utilizaban y hoy lo utilizan son generalmente personas interesadas en la Informática, pero no profesionales. En este lenguaje se va procesando cada instrucción mediante un "intérprete de BASIC", no existe un compilador que traduzca el lenguaje a código máquina, una vez establecido el programa. Es la razón por la que, aunque el tiempo de programación puede ser reducido, el tiempo de máquina es bastante mayor que con otros lenguajes.

El BASIC sigue en primera línea, y su éxito va parejo con el crecimiento de los microordenadores. Como ya no se trata de pagar la hora de un ordenador grande a precio alto, el consumo de tiempo de máquina pasa a tener menos importancia frente a la facilidad en la programación. La mayoría de los fabricantes de "micros" se han decidido por este lenguaje, que ha sido mejorado en muchas ocasiones. (Incluso existen versiones que pueden ser compiladas.) Desde luego, es el lenguaje más conocido por los aficionados.

No todos los lenguajes de programación son americanos. Rebajemos algo el número, y digamos que casi todos lo son. Por ejemplo, el PASCAL es un lenguaje didáctico europeo, y para más señas, suizo. Su creador, Niklaus Wirth, un apacible profesor de la Politécnica de Zurich, lo diseñó pensando fundamentalmente en sus alumnos. Además, este lenguaje es susceptible de "crecer", hasta alcanzar la "medida" o "talla" del usuario. Su nombre es un homenaje a Blaise Pascal, matemático y filósofo francés del siglo XVIII.

Otro lenguaje didáctico, encaminado a resolver problemas algebraicos y gráficos es el LOGO, que enseña al estudiante a mover una "tortuga" por la pantalla. La "tortuga" escribe líneas, figuras geométricas, etc. Fue crea-



do por Seymor Papert en la década de los setenta.

¿Sólo existen los lenguajes de programación antes mencionados? Desde luego que no: los humanos somos complejos, y existen casi tantos lenguajes como libros de cocina.

Para razonar la enorme proliferación de lenguajes, existen varias explicaciones, unas de orden comercial y otras de orden práctico.

En el orden comercial, ocurre como con los refranes, cada refrán tiene su contrarrefrán, y cada individuo puede elegir, según su situación, el que más le convenga. Así, "No por mucho madrugar amanece más temprano" y "A quien madruga Dios le ayuda", en el orden comercial se puede seleccionar entre fabricar "compatibles clónicos", o bien ordenadores cuyos compiladores lleven ciertas "particularidades especiales" que los hacen diferentes de otros ordenadores de la competencia. Evidentemente, estos ordenadores se han fabricado para que no puedan utilizar programas escritos en un lenguaje que es el que utiliza un equipo de la firma "enemiga". De este modo, los clientes siempre tienen que utilizar productos de esa misma casa. Este sistema es muy antiguo y, como saben los lectores, proporcionó enormes beneficios a la IBM en sus comienzos. (El negocio de las fichas constituía una partida importante para la empresa.)

La segunda razón se basa en la especialización. Los lenguajes especializados para temas determinados simplifican enormemente el trabajo de programación, y eso los hace más apreciados por el programador dedicado a tareas muy concretas.

Además de las razones antes citadas, también puede haber ocasiones en las que un determinado lenguaje ocupe mucha parte de la memoria del ordenador, mermando considerablemente sus posibilidades. La solución en estos casos consiste, pues, en utilizar otro lenguaje que tome menos memoria.

Existen muchos proyectos de lenguajes de programación universales. Que lo sean o no lo veremos en el futuro. Sin embargo, grandes constructores y usuarios de informática están interesados en proyectos de este tipo. Uno de ellos es el proyecto HOL (High Order Languaje) —Lenguaje del nivel superior—. Uno de sus principales promotores es el Departamento Norteamericano de Defensa. ¿Otra vez la Defensa? Efectivamente, como recordará el lector, desde los primeros tiempos de la Informática con el ENIAC y otros ordenadores utilizados con fines militares, o al menos defensivos, el Ministerio de Defensa Americano (entre otros) ha ido adquiriendo un enorme parque de ordenadores para sus muchos organismos. Sin embargo, cada ordenador se compraba para resolver un problema o problemas determinados, y de esto se deduce que el conjunto es absolutamente heterogéneo. Evidentemente, la unificación da como resultado un ahorro de medios, y ese ahorro puede ser considerable.

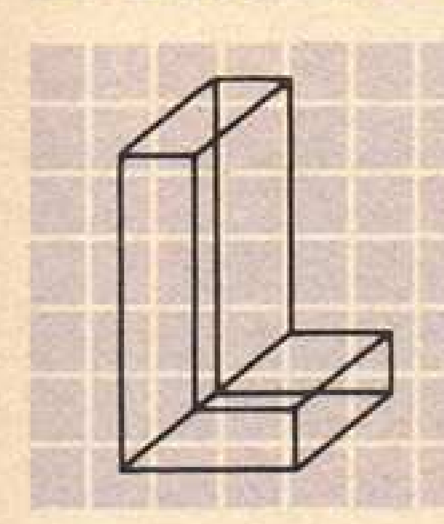
En 1977 el Departamento de Defensa de los Estados Unidos convocó un concurso mundial. Después de valorar comparativamente los numerosos proyectos de lenguajes universales recibidos, que habían elaborado las más prestigiosas firmas mundiales, el seleccionado fue el desarrollado por CII Honeywell Bull. Su nombre es ADA, en homenaje a Ada Lovelace, hija del poeta Byron y considerada como la primera programadora mundial.

Otros lenguajes de programación orientados a trabajos específicos son el LISP y el PLANNER, que se utilizan en Inteligencia Artificial, aunque el LIPS, por ejemplo, sea un lenguaje creado en 1958.

En Robótica, los lenguajes deben poder ser ejecutables en tiempo real, de forma muy similar a los utilizados en el control de procesos industriales. Por la propia naturaleza de los problemas a resolver, deben ser lenguajes de muy alto nivel. Entre ellos mencionamos el AL, basado en el PASCAL, y desarrollado en la Universidad de Stanford, y otros más desarrollados a partir de él como el VAL, etc.

Entre los lenguajes de propósito general no debemos olvidarnos del PL/1, diseñado para obtener las mejores características de los lenguajes COBOL y FORTRAN (se trataba de obtener un lenguaje que fuera útil en trabajos científicos y comerciales). No pensamos que este lenguaje haya tenido la repercusión que se esperaba.

Bases de datos relacionales



AS Bases de Datos Relacionales (BDR) forman un grupo dentro del conjunto general de los Sistemas de Gestión de Bases de Datos (SGBD) que, aunque definidas hace muchos años (1970-1972), han adquirido su desarrollo más modernamente y parece que se imponen, junto con otros modelos de datos (modelo entidad-relación, representación dirigida al objeto, etc.) como el esquema de representación y tratamiento de información de los 90.

Una Base de Datos (BD) es un conjunto de datos o informaciones sobre una realidad (una empresa u organización, o un área de conocimientos), almacenada (memorizada) por un ordenador, utilizada por numerosos usuarios y cuya organización está regida por un *modelo de datos.*

El informe ANSI-PARC de 1976 consideraba que una Base de Datos debe satisfacer cinco criterios:

1. Buena representación del mundo real.

— Imagen fiel de la realidad.
— Información fiable y actualizada.

2. Información no redundante.

— Cada dato debe aparecer una sola vez (situado en un solo sitio) dentro del conjunto de la BD.

3. Independencia de los programas de aplicación respecto de los datos.

4. Seguridad y confidencialidad de los datos.

— Seguridad física (contra la destrucción y contra la introducción de errores).
— Protección contra accesos indebidos por parte de personas no autorizadas.

5. Alta calidad de las aplicaciones y sistemas de consulta.

— Posibilidad de respuestas rápidas.
— Organización específica para mejorar los tiempos de tratamiento.

Se suelen definir tres niveles de modelización de los datos:

1. Nivel externo. Representa la visión de los datos que tiene el utilizador de los datos. Esta visión es, por su propia naturaleza, parcial e incompleta. Por otro lado, hay tantos modelos de datos como utilizadores: cada uno maneja solamente una parte de los datos y de un modo específico, aunque, sin embargo, haya zonas comunes en que los datos se solapan.

2. Nivel conceptual. Es la síntesis de todos los modelos externos. En teoría, se supone que tal modelo conceptual existe y, por tanto, que los modelos externos de cada utilizador son como versiones particulares de este modelo conceptual genérico.

3. Nivel físico. De hecho, los modelos conceptuales teóricos no pueden ser llevados a la práctica en una máquina, pues los equipos de que disponemos tienen limitaciones de tamaño, de capacidad de proceso, etc. De ahí surge la necesidad de estudiar cómo el SGBD almacena los datos y los procesa. No se dispone nunca del software necesario para que cree un diseño físico que se adecue al modelo conceptual definido, pero modernamente los SGBD son sumamente flexibles y permiten una gestión eficaz de los datos.

Históricamente, se suele hablar de tres generaciones de SGBD:

1.ª generación (finales de los sesenta).

Incluye los modelos básicos de gestión diseñados en un principio:

a) Modelos jerárquicos en que la estructura de los datos se expresa con la ayuda de una jerarquía arborescente.

El sistema más representativo de este grupo es el IMS (cuyo Lenguaje de Manipulación es el DL1).

b) Modelos en red. Desarrollados de acuerdo con la especificación CODASYL, permiten una clara separación entre la lógica y la física. En esta línea, los sistemas más representativos son el IDMS (IBM), el IDS-II (Honeywell) y SOCRATE (Bull).

2.ª generación (desde finales de los setenta).

Constituida por los modelos relacionales. En estos sistemas, la información se representa en forma tabular.

Existe una independencia total entre la lógica y la física. Hay cantidad de productos industriales disponibles en esta línea: ORACLE, FOCUS, INGRES, D-BASE III, etc.

3.ª generación (desde hoy en día).

Se están ampliando hoy los modelos de representación hacia estructuras más complejas, incluyendo el tratamiento de datos multimedia (sonido, imagen, texto), el tratamiento de bases de datos lógicos, bases de datos deducibles...

Hoy por hoy, este tipo de SGBD son esclusivamente experimentales.

Las bases de datos relacionales aportan un conjunto de características muy interesante:

a) Utilizan estructuras de datos muy simples (tablas) y que se adecuan al sistema de manejo de datos usual.

b) Facilitan a programadores y gestores de la Base, lenguajes de alto nivel (4.ª generación).

c) Posibilitan una gran independencia entre los datos y sus tratamientos.

d) Permiten modelos de utilizador diferentes de las relaciones básicas implantadas.

Las BDR se basan en tres elementos fundamentales: un soporte de la información (tabla), unas operaciones (relacionales o de conjuntos) y un estricto sistema de normalización.

1. El concepto fundamental en una BDR es la llamada "relación" de datos, plasmada en una "tabla". La tabla representa la información básica, organizada bajo unos epígrafes o atributos y sustituye a los "registros" de los sistemas tradicionales de soporte de la información.

Un ejemplo de tabla convencional es el siguiente:

N.º de producto	Nombre	Cantidad stock	Precio unitario	Precio total
102	Enchufe	50	318	15.900
105	Clavija	87	240	20.880
106	Múltiple	95	302	28.690
109	Interruptor	150	515	77.250

La tabla, como se ve, está organizada en filas u ocurrencias del registro (llamadas también "tuplas") y en columnas. En cada columna se agrupan todos los tipos de campo iguales de cada tupla (que responden a un determinado atributo: "nombre", "n.º de producto", etcétera). A cada campo (un determinado atributo —columna— de una determinada tupla —fila—) se le llama elemento. "Grado" de la relación es el número de columnas de la tabla. La "cardinalidad" representa el número de filas de la tabla.

Sin embargo, las tablas deben tener algunas condiciones especiales para ser manejadas adecuadamente en el sistema relacional:

— Cada tabla sólo puede tener un tipo de información (registro): sus filas y columnas deben ser uniformes. (Aunque la BDR, generalmente, tendrá muchas tablas.)

— No se permiten datos duplicados. De hecho, cuando aparece alguna fila o columna duplicada como resultado de una operación, el propio sistema la elimina.

— Los valores que toma cada campo (atributo) en una fila de la tabla deben limitarse a un conjunto concreto llamado "dominio".

— El orden en que aparecen los datos en la tabla es irrelevante.

2. Las operaciones de manejo de las tablas en un sistema de BDR son de dos tipos:

— Operaciones relacionales.

— Operaciones de conjuntos.

2.1. Operaciones relacionales son la restricción o selección, la proyección y la unión o concatenación.

— La restricción o selección consiste en la construcción de otra tabla, seleccionando algunas filas de la tabla original mediante un adecuado criterio de selección. Por ejemplo, mediante el criterio PRECIO < 305 se obtendría la tabla:

N.º de producto	Nombre	Cantidad stock	Precio unitario	Precio total
106	Múltiple	95	302	28.690
105	Clavija	87	240	20.880

(Obsérvese que el orden de las filas es irrelevante y el resultado estará ordenado como aparece en la tabla o en otro orden cualquiera.)

— La proyección es la construcción de otra tabla (a partir de la original) mediante la selección de ciertas columnas.

Una proyección sobre "valores totales", a partir de la tabla dada, originaría la siguiente tabla resultado:

N.º de producto	Nombre	Precio total
102	Enchufe	15.900
106	Múltiple	28.690
109	Interruptor	77.250
105	Clavija	20.880

— La unión o concatenación produce otra tabla relacionando dos atributos (uno de cada tabla) por coincidencia.

Así, por ejemplo, se puede concatenar la tabla dada con otra de "clientes de cada producto".

N.º de producto	Cliente	Población	Teléfono
105	Pérez	Madrid	345 67 12
105	López	Bilbao	415 34 77
109	Pérez	Madrid	345 67 12
106	Martínez	Sevilla	671 43 25
109	Martínez	Sevilla	671 43 25
102	López	Bilbao	415 34 77

Para dar la siguiente tabla:

N.º de producto	Nombre	Cantidad stock	Precio unitario	Precio total	Cliente	Población	Teléfono
105	Clavija	87	240	20.880	Pérez	Madrid	345 67 12
105	Clavija	87	240	20.880	López	Bilbao	415 34 77
106	Múltiple	95	302	28.690	Martínez	Sevilla	671 43 25
102	Enchufe	50	318	25.900	López	Bilbao	415 34 77
109	Interruptor	150	515	77.250	Pérez	Madrid	345 67 12
109	Interruptor	150	515	77.250	Martínez	Sevilla	671 43 25

2.2. Operaciones de conjuntos.

Normalmente, en los sistemas BDR se admiten también operaciones de conjuntos: unión, intersección y diferencia, entre las relaciones o tablas que forman la BDR. Además, es usual que se maneje el producto cartesiano (o una cierta restricción de él). Para todas estas operaciones, las tablas son consideradas como matrices a unir, intersecar, etc.

3. La normalización.

La presentación sin restricción alguna de las relaciones o tablas en un sistema de BDR llevaría a ciertas dificultades: podría aparecer información redundante, podría perderse información en alguna operación relacional, etcétera. Para evitarlo, se hace necesario normalizar las tablas de la BDR.

Producto	Desarrollado por:	Año
ADABAS	SOFTWARE AG	1972
DATACOM/DB	ADR	1974
CA-UNIVERSE	COMPUTER ASSOC.	1980
DG SQL	DATA GENERAL	1984
DMS 2	BURROUGHS	1972
DB 2	IBM	1983
IDMS/R	CULLINET	1972
INGRES	RTI	1980
MIMER	MIMER	1982
ORACLE	ORACLE	1979
RAPPORT	LOGICA	1976
RELATE 3000	CRI	—
SQL/DS	IBM	1980
TIS	CINCOM	1983
ULTRA	CINCOM	1983

(FUENTE: Guide Européen des Progiciels.) Sistemas de Gestión de Bases de Datos (SGBD) que se ofrecen en el mercado como "sistemas relacionales".

Producto	Eq. material (Hardware)	Logical (Software)
ADABAS	IBM 360 - 30XX Compatibles	DOS,MVS, VM, XA
DATACOM/BD	IBM 360 - 30XX Compatibles	DOS,MVS,VM XA
CA-UNIVERSE	IBM 370 - 30XX Compatibles	DOS,MVS,VM XA
DG SQL	ECLIPSE	AOS
DMS 2	B1000 - B7000 Serie A	MCP
DB 2	IBM 370 - 30XX	MVS,XA
IDMS/R	IBM 360 - 30XX Compatibles	DOS,MVS,VM XA
INGRES	DIGITAL VAX,IBM,... IBM 370 - 43XX,BULL,CDC, DIGITAL	VMS,UNIX,... MVS,VM,DOS VMS,NOS, GCOS
ORACLE	IBM 370 - 30XX, VAX, DG,...	VM,MVS,VMS,RSX AOS,UNIX,...
RAPPORT	IBM 370 - 30XX,VAX,BULL, PRIME	VM,MVS,VMS, GCOS 8,UNIX
RELATE 3000	HP 3000	MPE
SQL/DS	IBM 43XX - 30XX	VSE, VMS
TIS	IBM 370 - 30XX	DOS,MVS
ULTRA	DIGITAL VAX 730 - 780	VMS

Equipos (material y logical) sobre los que están implementados los principales SGBD relacionales.

Se han establecido tres formas normales para las relaciones (tablas) de las BDR (las formas normales 4.ª y 5.ª son muy específicas y no merece la pena detallarlas).

Una relación está en primera forma normal (PFN) si no incluye ningún grupo repetitivo. Además, se cumple que en una relación que esté en PFN los dominios de sus atributos no tienen elementos que sean, a su vez, conjuntos. Una relación está en segunda forma normal (SFN) si, estando en PFN, todos sus atributos dependen de la clase completa y no sólo de una parte de ella.

Por último, la tabla estará en tercera forma normal si está en SFN y, además, todas las dependencias funcionales establecidas lo son respecto de la clave; es decir, si todos los atributos de la relación dependen sólo de la clave y no de algún otro atributo.

Hay que reseñar, por último, que existen desarrolladas las llamadas máquinas de Bases de Datos (MBD). Son, en esencia, máquinas diseñadas especialmente para soportar las estructuras y los modos de trabajo de los SGBD. Normalmente, las MBD son el centro de algún sistema de ordenadores más sofisticados y están, por tanto, "rodeadas" de otros equipos que las auxilian: estos otros ordenadores reciben las peticiones de información del usuario y las transcriben a los lenguajes propios del SGBD; y reciben las respuestas y las preparan para su adecuada presentación al usuario. Entre las MBD hay algunas (ADABAS, por ejemplo), que se soportan en equipos convencionales (o compatibles) comerciales; otras (IDM 500, por ejemplo) son máquinas específicas cuya arquitectura está especialmente pensada con esta finalidad.

TERMINOLOGIA

GLOSARIO DE TERMINOS UTILIZADOS CON LAS BASES DE DATOS RELACIONALES

Atributo. Cada uno de los "campos" o apartados de que consta cada anotación (ocurrencia o tupla) que aparece en una tabla. Los atributos de cada tupla aparecen en columnas encabezadas por el nombre del atributo.

BD. Se utilizan estas siglas para referirse a las Bases de Datos.

BDR. BD relacional.

Campo. Cada elemento de la tabla. Se utiliza más este nombre por similitud con los componentes de un registro (asimilable a una fila de la tabla) de los sistemas convencionales de almacenamiento y recuperación de datos.

Cardinalidad. Cardinalidad es el número de filas o tuplas que aparecen en una tabla representativa de una relación.

Concatenación. Una de las operaciones relacionales. Consiste en la reunión de dos tablas o relaciones a través de un atributo común, basándose en la similitud de valor de dicho atributo en las diferentes tuplas de la relación.

Diferencia. Operación de conjuntos. El resultado de la diferencia de los conjuntos A y B (A–B) es el conjunto de los elementos de A que no están en B.

Dominio. También llamado "dominio de discurso". Conjunto de los valores que puede adoptar una variable. En la teoría relacional se utiliza para definir el ámbito de definición de los valores que puede adoptar cada atributo.

Elemento. Campo que aparece en el cruce de una fila (o tupla) y una columna (o atributo) de una tabla.

Formas normales. Presentación especial de las tablas representativas de las relaciones en una BD relacional. Se utilizan las formas normales para definir las relaciones, con el fin de evitar duplicidades, pérdida de información, etc. Suelen utilizarse tres formas normales (1.ª a 3.ª) llamadas PFN, SFN, TFN.

Grado. Grado es el número de columnas de la tabla que representa una relación. Coincide con el número de atributos de la tabla, que es igual al número de elementos de cada tupla.

Intersección. Operación de conjuntos. La intersección de los conjuntos A y B está formada por el conjunto de los elementos que pertenecen simultáneamente a A y a B.

MBD. Máquina de Base de Datos. Estructura (física) de ordenador(es) electrónicos para la más eficaz implementación de un sistema de Gestión de Bases de Datos (SGBD). Es un computador específicamente diseñado con esta finalidad.

Modelo. Paradigma de representación del conocimiento para la adecuada estructuración de los datos. En los SGBD, en general, se utilizan modelos jerárquicos, en red, relacionales, de entidad-relación, basados en clases-objeto, etc. Cada modelo de representación suele llevar emparejado un lenguaje adecuado para el tratamiento de los datos.

Objeto, representación dirigida al. Modelo de representación de la información muy utilizada modernamente. Existen traba-

jos que reúnen las ideas del modelo objeto con el modelo relacional para su mejor eficacia.

Ocurrencia. Cada aparición de una unidad de datos en una relación. Aparece representada como una fila en la tabla correspondiente a dicha relación.

Producto cartesiano. Operación de conjuntos. El producto cartesiano de los conjuntos se construye combinando los elementos de uno de los conjuntos con los del otro. En los diferentes SGBD relacionales se suelen facilitar herramientas para realizar algunas variantes del producto cartesiano.

Proyección. Operación relacional. Consiste en la obtención de una tabla reducida a partir de otra dada, por extracción de ciertas columnas (atributos) de ella.

Registro. Unidad de información de un fichero de datos. En las BD relacionales se da este nombre a cada una de las ocurrencias de un tipo de datos (o fila de la tabla).

Relación. Elemento básico de una BD relacional. Es el conjunto homogéneo de datos existente sobre una materia. Viene representado por una tabla de filas y columnas. Normalmente se utilizan de un modo indistinto los nombres de tabla y relación.

Restricción. Operación relacional. También se llama selección. Ver **Selección.**

Selección. Operación relacional. Consiste en la creación de una tabla reducida (a partir de una dada) por extracción de algunas filas (o tuplas) de la primitiva, de acuerdo con un criterio dado.

SGBD. Sistema de Gestión de Bases de Datos. Conjunto formado por el modelo de representación de los datos, las herramientas para su implementación y un lenguaje para su fácil manejo por los usuarios.

Tabla. Plasmación gráfica de una relación, en un sistema relacional. A veces se usa en vez de relación. Ver **Relación.**

Tupla. Cada una de las ocurrencias (filas o registros) de una tabla representativa de una relación. Tiene tantos elementos como atributos se consideren en la relación. Su longitud (en campos) define el grado de la relación.

Unión. Operación relacional. Se da a veces el nombre de unión a la concatenación. Ver **Concatenación.**
— Operación de conjuntos. Unión o reunión de los conjuntos A y B, es el conjunto formado por todos los elementos que están en A o en B, o en ambos.

VOCABULARIO DE INFORMATICA

Búsqueda. Localización de un dato. Generalmente se refiere a la localización de un registro en un fichero, o un sector o una pista en un disco.

Búsqueda binaria. Método de búsqueda que se aplica a ficheros clasificados. El sistema consiste en ir efectuando divisiones en dos partes iguales, y buscar el elemento de que se trate en una de ellas. Una vez localizado en una de las partes, ésta se vuelve a dividir y se repite la operación. Es un método de búsqueda relativamente rápido.

Búsqueda dicotómica. (Ver **Búsqueda binaria.**)

Búsqueda secuencial. Método de búsqueda que consiste en inspeccionar los elementos del fichero partiendo del primero hasta alcanzar el elemento deseado. Es un sistema muy lento, pero en ciertos casos, si el archivo está estructurado en forma secuencial, es el único posible.

Byte. Conjunto de bits a los que se considera formando una unidad. Normalmente son más cortos que una palabra. Habitualmente se considera el byte de 8 bits.

Byte por segundo. Medida de la velocidad de una transmisión. Se refiere al número de bytes transferidos en un segundo.

Cabecera. *a*) Encabezamiento de un impreso o informe. *b*) Información que se da al contenido de un bloque de datos.

Cabeza. *a*) Puede referirse a una cabeza de grabación magnética (diskette, disco duro o cassette) para escribir datos en esos dispositivos o leerlos. *b*) A la cabeza de escritura de una impresora.

Cabeza de lectura/escritura. Dispositivo consistente esencialmente en unos bobinados sobre núcleos especiales, dedicada a leer o escribir datos. (Véase **Cabeza.**)

Cabrestante. Polea muy especializada y precisa que arrastra la cinta magnética (en unidades de cinta o cassettes). Su precisión es fundamental para la grabación y posterior recuperación de los datos. (Los carretes donde se arrolla la cinta no controlan en absoluto el movimiento de ésta.)

Caché, memoria. Memoria inmediata. Se utiliza como memoria de almacenamiento intermedio en ordenadores grandes, generalmente. Sirve para reducir tiempos de acceso a la memoria central.

CAD (Computer Aided Design). Diseño Asistido por Ordenador. Paquetes gráficos en los que el diseño se realiza con un terminal de ordenador. Los dibujos se almacenan en ficheros recuperables para ser mejorados, etc. También disponen de algunos programas de cálculo. En resumen, constituyen una ayuda completa para el proyectista, realizando tanto cálculos como gráficos. (Véase **CAM** y **CAD/CAM.**)

CAD (Conversor Analógico Digital). Dispositivo que convierte señales analógicas (o continuas) en digitales (o discontinuas). Existe otro dispositivo que realiza la acción inversa, el Conversor Digital/Analógico.

CAD/CAM. Sistema integrado que reúne un paquete de diseño asistido (CAD) y otro de fabricación asistida por ordenador (CAM).

Cadena. Sucesión de caracteres que se toma como unidad para utilizarla en algún proceso.

Cadena, búsqueda de. Técnica en la que cada elemento contiene un identificador que permite la localización del siguiente elemento.

Cadena, lista, listado. Lista de elementos que pueden estar dispersos, en la cual cada uno de ellos contiene un identificador que permite la localización del siguiente elemento a considerar.

Cadencia. Demora producida por la ejecución de una operación de entrada/salida, y almacenaje o presentación de un dato.

CAE (Computer Aided Education). Enseñanza Asistida por Ordenador. Paquetes de enseñanza interactiva en las que el ordenador colabora con el profesor. Por ejemplo, para aprender a escribir a máquina el alumno va escribiendo un texto que aparece en la pantalla. Terminado éste, el ordenador cuenta el número de palabras y el número de fallos. Si el alumno ha superado ese nivel, pasa al nivel siguiente. Si no lo ha superado, generalmente se repite la secuencia.

CAL (Computer Aided Learning). Aprendizaje Asistido por Ordenador. Semejante a CAE. (Véase **CAE.**)

Calculadora. *a*) Dispositivo de proceso de datos diseñado especialmente para realizar operaciones aritméticas. Normalmente suele requerir la intervención frecuente de un operador. *b*) Dispositivo que lleva a cabo todo tipo de operaciones digitales, aritméticas y lógicas.

Call (llamada). Se utiliza: *a*) Para transferir el control a una subrutina específica. *b*) En comunicaciones, operaciones necesarias para conectar dos estaciones. Utilización del canal de conexión entre ellas.

CAM (Computer Aided Manufacturing). Fabricación Asistida por Ordenador. Sistema informático de planificación y control de fabricación. Suele estar integrado con un sistema de diseño asistido, formando un CAD/CAM. (Véase **CAD.**)

CAN. Se refiere al carácter de cancelar, CANCEL.

Camino. *a*) Especificación completa de un fichero dentro de un subdirectorio en el sistema operativo MS-DOS. *b*) Vía lógica para la transferencia de datos. Sinónimo de canal lógico.

Campo. Parte de un registro que contiene un dato individual. Por ejemplo, un fichero de una biblioteca tiene registros, cuyos campos pueden ser los siguientes: título, autor, tipo de libro, fecha de publicación, etc.

Canal. *a*) En transmisión de datos, vía de comunicación entre la unidad central y los periféricos, tanto en sentido físico como lógico. *b*) Zona de un dispositivo de almacenamiento que es accesible a determinada estación de lectura o escritura.

Capstan. Dispositivo de las unidades de cinta magnética de los ordenadores electrónicos. (Véase **Cabrestante.**).

Carácter. Letra, dígito u otro símbolo cualquiera que se utiliza en la organización, control o representación de los datos. Pueden ser las letras del alfabeto, los números, signos de puntuación, caracteres especiales de control, etc.

Carácter de control. Carácter perteneciente al conjunto de caracteres de un determinado equipo, que produce un efecto especial determinado.

Carácter, frontera de. En reconocimiento de caracteres, el mayor rectángulo uno de cuyos lados sea paralelo al borde de referencia del documento y cuyos otros dos lados sean tangenciales a una línea de caracteres especificada.

Caracteres gráficos. Subconjunto del juego general de caracteres utilizado específicamente en la creación de gráficos. Son utilizados en paquetes gráficos y sirven para ir componiendo los gráficos y dibujos, colocándolos uno junto a otro, etc. El ordenador los considera y utiliza como si fueran cadenas de caracteres normales.